NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DEVELOPMENT OF A MYKLESTAD'S ROTOR BLADE DYNAMIC ANALYSIS CODE FOR APPLICATION TO JANRAD

by

Dogan Ozturk

September 2002

Thesis Advisor:	E. Roberts Woods
Co-Advisor :	Mark A. Couch

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Development of a Myklestad's Rotor Blade Dynamic Analysis Code for Application to Janrad	**5. FUNDING NUMBERS**
6. AUTHOR(S) Dogan Ozturk	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT

Blade Dynamics Analysis is a major portion of a helicopter design. The success of the design is strictly related to accuracy of the blade dynamic calculations. The natural frequencies and the mode shapes are not only difficult to calculate but also can be a time consuming procedure. The Myklestad Extension Method gives the designer the opportunity of calculating correct values of the natural frequencies and the mode shapes when centrifugal forces are present. This thesis provides a transfer matrix Myklestad analysis programmed in MATLAB® and a Graphical User Interface (GUI) tool built in the MATLAB® programming language version 6.1, to implement the Myklestad Extension Method. The generated code and the GUI are designed to be a part of 'Blade Dynamics Module' of Joint Army/Navy Rotorcraft Analysis Design (JANRAD). For comparison, nonrotating and uniform beam data from Young and Felgar and the actual data of an H-3 (S-61) helicopter blade are used. Results of the comparison show that the accuracy and robustness of the program are very good, which would make this generated code a valuable part of the helicopter designer's toolbox.

14. SUBJECT TERMS Rotor Blade Dynamics, Myklestad Extension Method, JANRAD, MATLAB, GUI. Natural frequencies, mode shapes, flatwise axis, edgewise axis.	**15. NUMBER OF PAGES** 236
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

DEVELOPMENT OF A MYKLESTAD'S ROTOR BLADE DYNAMIC ANALYSIS CODE FOR APPLICATION TO JANRAD

Dogan Ozturk
First Lieutenant, Turkish Army
B.S., Turkish Army Academy, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author: Dogan Ozturk

Approved by: E. Roberts Wood
Thesis Advisor

Mark A. Couch
Co-Advisor

Max F. Platzer
Chairman, Department of Aeronautics and Astronautics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Blade Dynamics Analysis is a major portion of a helicopter design. The success of the design is strictly related to accuracy of the blade dynamic calculations. The natural frequencies and the mode shapes are not only difficult to calculate but also can be a time consuming procedure. The Myklestad Extension Method gives the designer the opportunity of calculating correct values of the natural frequencies and the mode shapes when centrifugal forces are present. This thesis provides a transfer matrix Myklestad analysis programmed in MATLAB® and a Graphical User Interface (GUI) tool built in the MATLAB® programming language version 6.1, to implement the Myklestad Extension Method. The generated code and the GUI are designed to be a part of 'Blade Dynamics Module' of Joint Army/Navy Rotorcraft Analysis Design (JANRAD). For comparison, nonrotating and uniform beam data from Young and Felgar and the actual data of H-3 (S-61) helicopter blade are used. Results of the comparison show that the accuracy and robustness of the program are very good, which would make this generated code a valuable part of the helicopter designer's toolbox.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF SYMBOLS AND ABBREVIATIONS

β	Blade Flap Angle
β_T	Twist Angle
Ω	Rotor Angular Velocity
ω	Natural Frequency
ρ	Density of Air
θ	Blade Pitch Angle
η	Discontinuity
ξ	Blade Lag Angle
C_n	Flatwise Aerodynamic Damping
Dn+jdn	Aerodynamic Drag Force Acting on the Blade Elements
e	Hinge offset
$F_n + jf_n$	Aerodynamic Lift Force Acting on the Blade Elements
g	Deflection Due to Load About Flatwise Axis (Coupling Term)
G	Deflection Due to Load About Edgewise Axis (Coupling Term)
I_β	Static Flapping Moment of Blade
I_ξ	Static Lagging Moment of Blade
l	Length of the section
m	Mass of the Blade Element
M	Bending Moment at the Blade Station
r	Rotor Blade radius
S	Shear Force Acting on the Blade Element
S_β	Mass Flapping Moment of Inertia
S_ξ	Mass Lagging Moment of Inertia
T	Centrifugal Tension Force Acting on the Blade Element
X	Edgewise Displacement of the Blade Station
u	Slope Due to Load About Flatwise Axis (Coupling Term)
U	Slope Due to Load About Edgewise Axis (Coupling Term)
v	Slope Due to Moment About Flatwise Axis (Coupling Term)
V	Slope Due to Moment About Edgewise Axis (Coupling Term)
Z	Flatwise Displacement of the Blade Station
$-^E$	Denotes Edgewise
$-^F$	Denotes Flatwise
$-_0$	Denotes Root
$-_T$	Denotes Tip
AHS	American Helicopter Society
c.g.	Center of Gravity
GUI	Graphical User Interface
JANRAD	Joint Army/Navy Rotorcraft Analysis and Design
UAV	Unmanned Aerial Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank to my advisors, Professor Wood and Commander Couch for their time, patience, and contributions. It was an honor for me to work with them.

I would also like to thank my family for their support from my country. With all my love and respect, I dedicate this thesis to my parents, Fikri and Done Ozturk.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) computer code was originally developed by students at the Naval Postgraduate School (NPS) for use in the 1993 American Helicopter Society (AHS) Design Competition. JANRAD was developed to include three primary modules. These modules are Performance, Stability and Control, and Structural Dynamics.

The computer code for interactive rotorcraft preliminary design using a harmonic balance method for rotor trim was developed by Nicholson [Ref. 1]. The linear modeling of Stability Analysis and preliminary design was developed by Wirth [Ref. 2] in 1993. Cuesta [Ref. 3] used a modified Myklestad-Prohl transfer matrix method for modeling helicopter blade dynamics. Further improvements to the rotor dynamics portion were developed by Hiatt [Ref. 4], followed by a validation of the JANRAD by comparing with H-34 and UH-60A flight test data was made by Eccles [Ref. 5]. Klein [Ref. 6] developed linear modeling of tilt rotor aircraft (in helicopter and airplane models) for stability analysis and preliminary design. Lapacik [Ref. 7] developed the Graphical User Interface (GUI) for JANRAD, and Hucke [Ref. 8] made the performance enhancements to JANRAD and GUI. McEwen [Ref. 9] utilized JANRAD to determine the stability and control derivatives used in the simulation model of a small rotary wing UAV. And most recently Heathorn [Ref. 10] developed the stability and control module for JANRAD software and GUI. These codes have been used efficiently in helicopter design courses at the Naval Postgraduate School and for AHS Design Competitions.

The need of detailed and organized analysis on blade dynamics in JANRAD was basic inspiration of this research. The MATLAB® code generated in this thesis is going to be one part of the improvement to the blade dynamics analysis module of JANRAD and is focused on finding the mode shapes of the helicopter blades.

B. INTRODUCTION TO ROTOR BLADE DYNAMICS

Dynamics is the study of how things change with time, and of the forces that bring these changes about. The forces that change with time and act upon the rotor blade are aerodynamic forces. Application of these time-varying forces cause a dynamic response to the blade with resultant forces at the blade root which result in an associated dynamic response to the airframe and its components. In addition, the helicopter incorporates an entire system of dynamic components consisting of the engines, engine gearboxes, coupling drive shafts, the main transmission, tail rotor drive shaft, intermediate and 90-degree tail rotor gearboxes which also cause dynamic response to the airframe [Ref. 11]. The focus of this thesis will be on rotor blade dynamics.

The modern helicopter has reached the present state of design without waiting for a complete definition of its aerodynamic and dynamic characteristics. Instead, it has been designed by extrapolation of existing parameters, and then allowed to prove itself through flight test and development programs [Ref. 12].

A general helicopter aeromechanics problem can be subdivided into five major categories, these are [Ref. 12]:

- Air Mass Dynamics (see Figure 1)
- Calculation of Aerodynamic Loads (see Figure 2)
- Rotor Blade Dynamics (see Figure 3)
- Blade-Fuselage Coupling (see Figure 4)
- Fuselage Dynamics (see Figure 5)

Throughout the history of helicopter design studies, all major areas of the aeromechanics problem were not able to be solved at the same time. Design studies were normally focused on one specific area or at most two of the areas. In recent years, the increased capability of the computer has allowed the designer to look at all areas simultaneously.

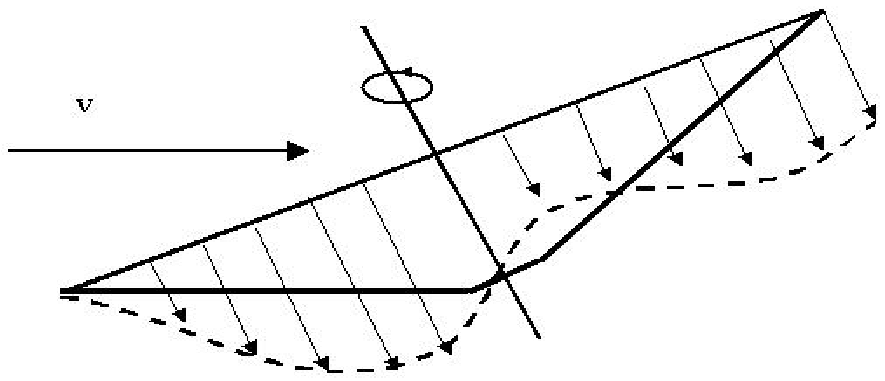


Figure 1. Air Mass Dynamics “From [Ref. 12]”

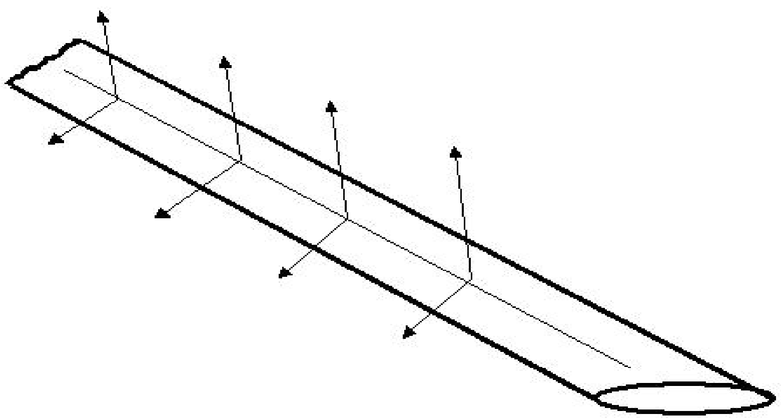


Figure 2. Calculation Of Aerodynamic Loads “From [Ref. 12]”

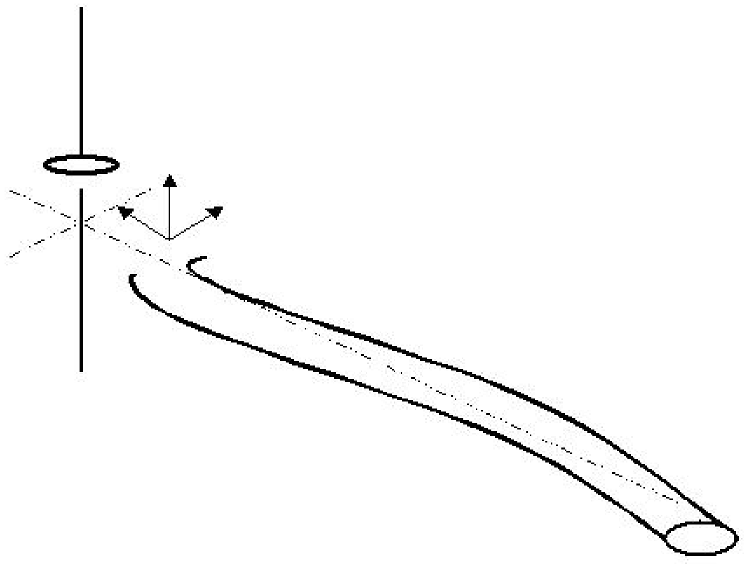


Figure 3. Rotor Blade Dynamics “From [Ref. 12]”

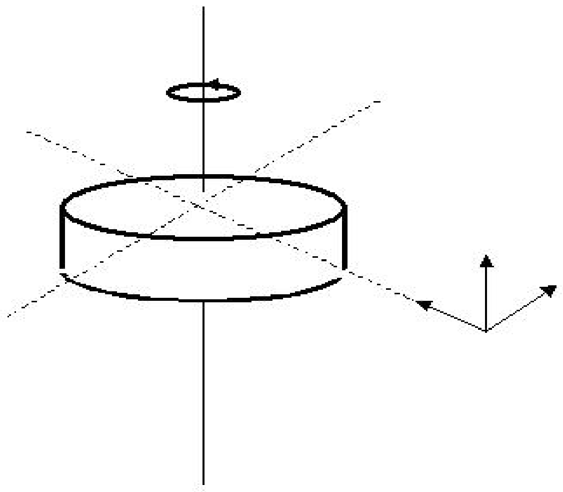


Figure 4. Blade-Fuselage Coupling “From [Ref. 12]”

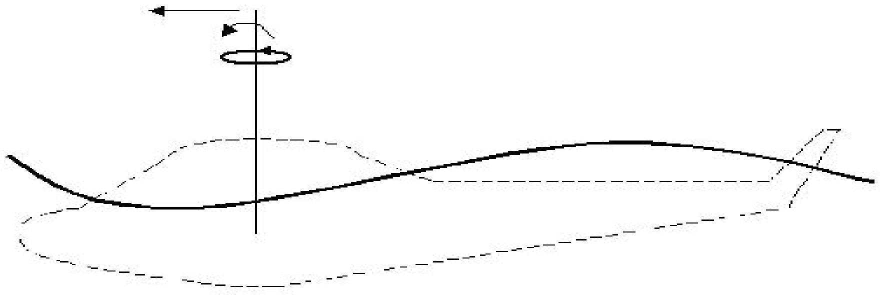


Figure 5. Fuselage Dynamics "From [Ref. 12]"

Rotor Blade Dynamics, one of the five categories of aeromechanics problem above, plays a major role in design and development of the modern-day helicopter. For that reason, the success of the design or development of a helicopter is directly affected by the dynamics of the rotor blades. As in many physical systems, the helicopter blade has a number of natural frequencies. The number of frequencies depends upon the number of coordinates necessary to define the motion of a blade. Natural frequencies have significant importance because if the rotor blade system is excited at or close to one of its natural frequencies, the resulting forces and motions will be amplified. For that reason a primary goal in a good helicopter design is to have the natural frequencies sufficiently distant from known excitation frequencies.

The rotating helicopter blades have both flapping and lagging movements. During the flapping or lagging, the blades have an inertia force and a centrifugal force that act as spring. The Flapping blade (see Figure 6) and lagging blade (see Figure 7) and the related equations are listed below. flapping blade motion in Figure 6 is an out-of-plane motion while the lagging blade in Figure 7 is in-plane motion.

1. Flapping Blade

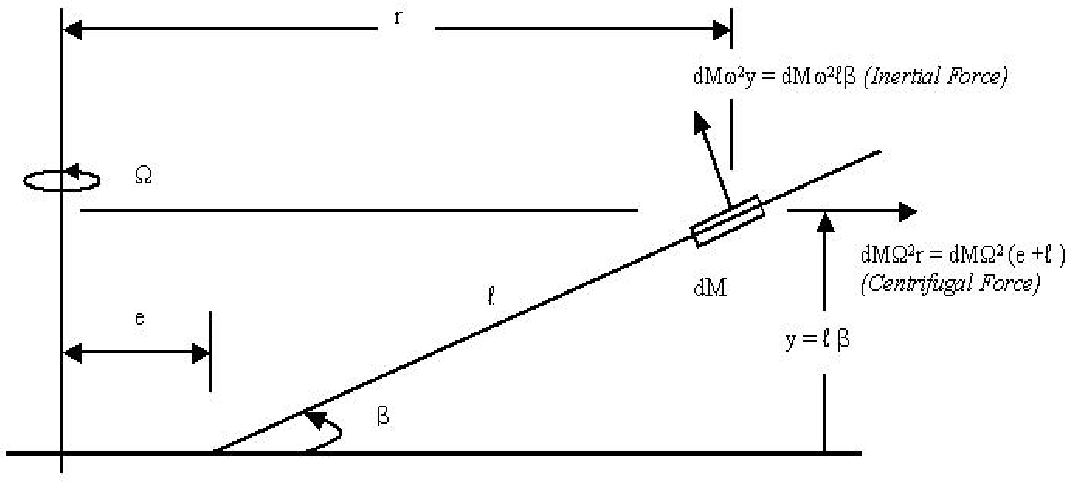

r
$dM\omega^2 y = dM\omega^2 \ell\beta$ (Inertial Force)
Ω
$dM\Omega^2 r = dM\Omega^2 (e + \ell)$ (Centrifugal Force)
ℓ
dM
e
$y = \ell\beta$
β

Figure 6. Flapping Blade "After [Ref. 11]"

If we assume β is a small angle:

$$\sin\beta = \beta \text{ and } \cos\beta = 1$$

Therefore;

$$\sum M_0 = 0 = \omega^2\beta \int_e^R \ell^2 dM \Omega^2\beta \int_e^R (e+\ell)\,\ell dM \tag{1}$$

Let

$\int_e^R \ell^2 dM = I_\beta$ Static flapping moment of blade,

$\int_e^R \ell dM = S_\beta$ Mass flapping moment of inertia

Hence;

$$\omega^2 I_\beta - \Omega^2 e S_\beta - \Omega^2 I_\beta = 0 \tag{2}$$

And;

$$\omega = \Omega\sqrt{1+\frac{eS_{\beta}}{I_{\beta}}} \tag{3}$$

If offset is equal to zero $e=0$;

Then, $\omega=\Omega$;

For normal offset $\omega=1.02\Omega$.

2. Lagging Blade

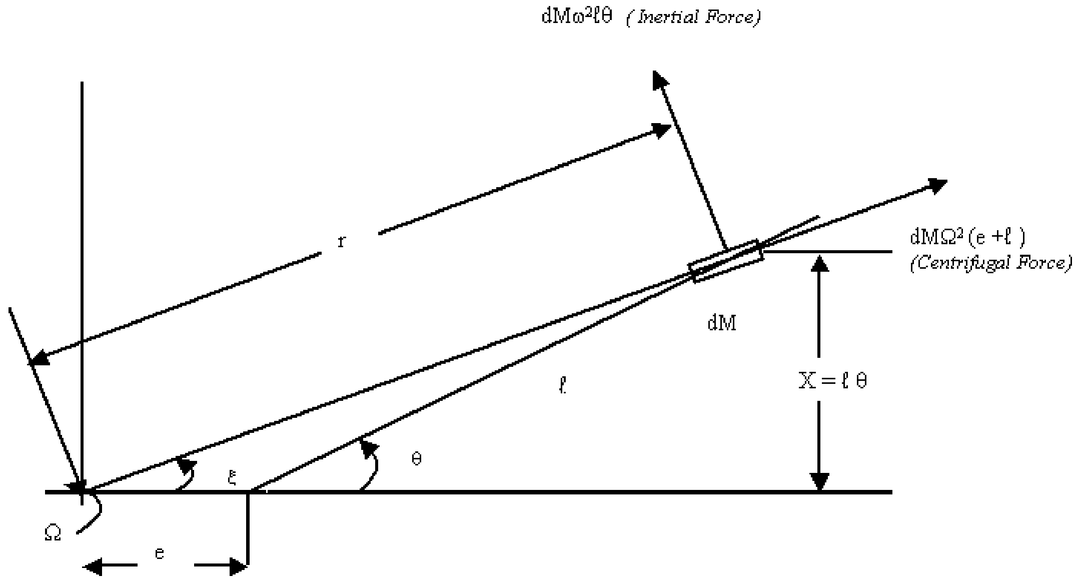


Figure 7. Lagging Blade "After [Ref. 11]

Assuming small angles;

$$\sin\xi=\xi \text{ and } \cos\xi=1$$

Therefore;

$$\sum M_0 = 0 = \int_e^R dM\omega^2\ell^2\theta + \int_e^R dM\Omega^2(e+\ell)\xi\ell - \int_e^R dM\Omega^2(e+\ell)\ell\theta = 0 \tag{4}$$

$$dM\Omega^2\xi = dM\Omega^2\theta\frac{\ell}{\ell+e} \tag{5}$$

$$\omega^2\theta\int_e^R \ell^2 dM + \int_e^R \Omega^2\frac{(e+\ell)\theta\ell^2}{(e+\ell)}dM - \int_e^R \Omega^2(\ell+e)\ell\theta dM = 0 \tag{6}$$

$$\omega^2\theta\int_e^R \ell^2 dM - e\Omega^2\theta\int_e^R \ell dM = 0 \tag{7}$$

Let

$\int_e^R \ell^2 dM = I_\xi$;

$\int_e^R \ell dM = S_\xi$;

Then;

$$\omega = \Omega\sqrt{\frac{eS_\xi}{I_\xi}} \tag{8}$$

For typical rotors; $\frac{\Omega}{4} \le \omega \le \frac{\Omega}{3}$.

For the case of multiple degrees-of-freedom the same principles apply as in the single degree-of-freedom case. The only difference is that there are additional natural frequencies for each additional degree-of-freedom. At each natural frequency (ω) the system vibrates in a characteristic shape. These characteristic shapes are called mode shapes. Every mode shape has maximum and minimum points. Starting from the second modes, mode shapes will also have zero amplitude at some points (referred to as nodes). The second mode shape has only one node. In general the nth node will have 'n-1' nodes (see Figure 8). For the system in Figure 8 $\omega_3 > \omega_2 > \omega_1$ [Ref. 11]. The natural frequencies are called eigenvalues and the mode shapes are eigenvectors. The statements in this paragraph are valid for all vibrating systems.

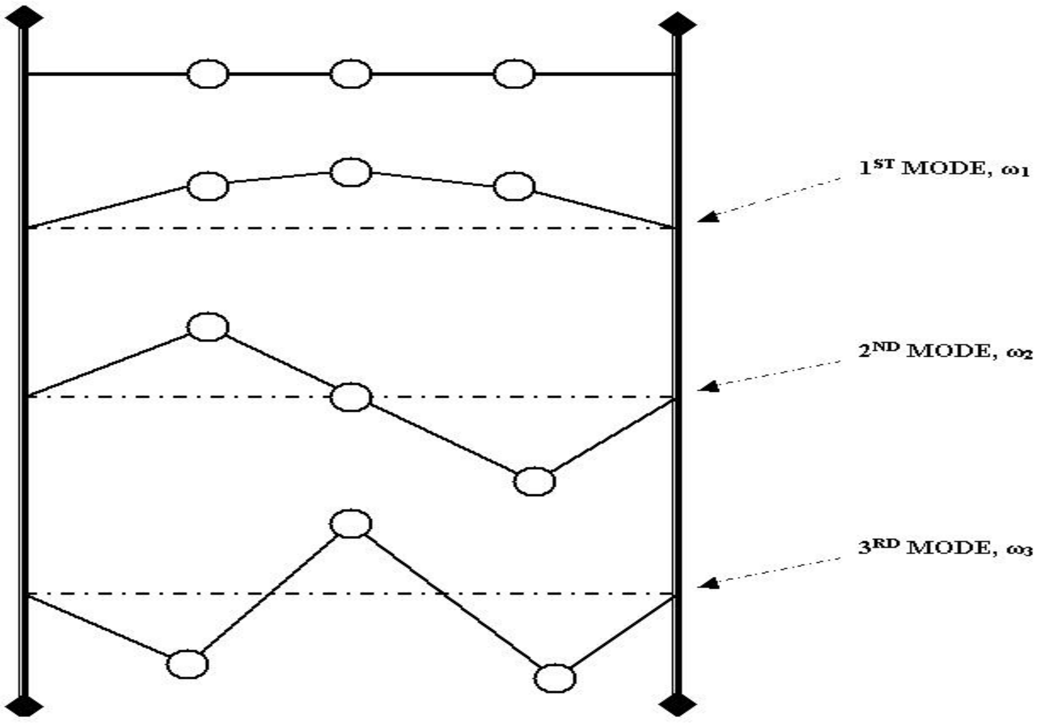


Figure 8. String With 3 Masses; (3 Degree of Freedom or Three Mode Shapes) “After [Ref. 11]”

THIS PAGE INTENTIONALLY LEFT BLANK

II. MYKLESTAD ANALYSIS

Vibrations of continuous systems, such as beams, helicopter fuselage, or rotor blades, can be analyzed mathematically by reducing the system to a system of discrete masses and springs. Some of the methods for analyzing vibrations are [Ref. 13]:

- Rayleigh's Energy Method
- Rayleigh-Ritz Method
- Stodola Method
- Holzer
- Myklestad
- Influence Coefficients Employing Matrix Methods

One method for determining the natural frequencies or critical speeds of shafts or beams in bending is the iteration method of Stodola, either in its graphical form or its numerical form [Ref. 14]. Also a useful and practical method of mode shape and frequency analysis was that developed by Holzer and has been used by engineers for years. The Holzer Method was developed primarily for application to torsional vibration problems [Ref. 15]. An extension of Holzer's method to the beam lateral vibration (or bending) problem was made by Myklestad and has proven useful in analysis of airplane wings, rotor blades and turbine blades. This method is referred to as Myklestad Extension or Myklestad-Prohl Method. Both Holzer's original method and Myklestad's Extension are effectively step-by-step solutions of the differential equation of a lumped parameter system.

A. THE DISCUSSION OF THE MYKLESTAD METHOD

The beam in question is first divided into a convenient number of sections. The mass of each section is calculated, divided into halves, and these halves concentrated at the two ends of each section. Thus the beam is weightless between cuts and at each cut there is a concentrated mass equal to half of the sum of the masses of the two adjacent sections. As in the Holzer method, we assume a frequency and proceed from section to section along the beam. In the torsional problem (governed by a second-order differential equation) there are two quantities of importance at each cut: the angle and the twisting moment. In the flexural problem (governed by a fourth-order differential equation) there are four quantities of importance at each cut: the deflection, the slope, the bending moment and the shear force. In order to solve either problem, it is necessary to find the relations between these quantities from one cut to the next. [Ref. 14]

In the flexural problem, if the shear, bending moment, slope and deflection at one station is known, it is possible to compute the corresponding values at the next station. To solve the problem, a value for a frequency is assumed, and the shear, bending moment, slope and deflection are calculated down the beam from mass to mass until the corresponding quantities at the other end are obtained. If the frequency chosen is not the correct one, boundary conditions at both ends of the beam will not be satisfied simultaneously.

In the past, the difficulty of this method was obtaining an accurate solution with a small number of stations. More stations mean the more iterations and the calculation of these values by hand was difficult and time consuming. Recently computer technology has made this method more attractive. The number of iterations can be increased to a larger number so that the accuracy can be increased. Twenty-five to fifty stations gives a close value to the exact solution.

B. MYKLESTAD METHOD APPLIED TO THE ROTOR BLADE

In actuality, the blade is a twisted beam. Its proper analysis requires considering the coupled flatwise-edgewise-torsional response of the blades. Slope and deflection relationship of the blade and resultant twist coupling terms can be derived as shown in the equations below. The blade has slope due to moment (v), slope due to load (u) and deflection due to load (g) values about its flatwise (flapwise) principal axis. Also, slope due to moment (V), slope due to load (U) and deflection due to load (G) are the values about the blade's edgewise (chordwise) principal axis. Allowing one discontinuity η in a station length $l_{n,n+1}$, we can obtain

$$v_{n,n+1} = \left[\frac{\eta_{n,n+1}}{(EI_y)_{n+1}} + \frac{l_{n,n+1} - \eta_{n,n+1}}{(EI_y)_n} \right] \tag{9}$$

$$u_{n,n+1} = \frac{1}{2} \left[\frac{\eta^2_{n,n+1}}{(EI_y)_{n+1}} + \frac{l^2_{n,n+1} - \eta^2_{n,n+1}}{(EI_y)_n} \right] \tag{10}$$

$$g_{n,n+1} = \frac{1}{3} \left[\frac{\eta^3_{n,n+1}}{(EI_y)_{n+1}} + \frac{l^3_{n,n+1} - \eta^3_{n,n+1}}{(EI_y)_n} \right] \tag{11}$$

$$V_{n,n+1} = \left[\frac{\eta_{n,n+1}}{(EI_x)_{n+1}} + \frac{l_{n,n+1} - \eta_{n,n+1}}{(EI_x)_n} \right] \tag{12}$$

$$U_{n,n+1} = \frac{1}{2} \left[\frac{\eta^2_{n,n+1}}{(EI_x)_{n+1}} + \frac{l^2_{n,n+1} - \eta^2_{n,n+1}}{(EI_x)_n} \right] \tag{13}$$

$$G_{n,n+1} = \frac{1}{3} \left[\frac{\eta^3_{n,n+1}}{(EI_x)_{n+1}} + \frac{l^3_{n,n+1} - \eta^3_{n,n+1}}{(EI_z)_n} \right] \tag{14}$$

Then if we take the blade element feathered at an angle, β, to the reference plane at right angles to the axis of rotation. Here it includes twist β_T and blade feathering θ. Then the slope and deflection relationships can be written as follows:

$$v_{zz} = V_{n,n+1} \sin^2 \beta + v_{n,n+1} \cos^2 \beta \tag{15}$$

$$V_{xx} = V_{n,n+1} \cos^2 \beta + v_{n,n+1} \sin^2 \beta \tag{16}$$

$$v_{zx} = V_{xz} = (V_{n,n+1} - v_{n,n+1}) \sin \beta \cos \beta \tag{17}$$

$$u_{zz} = U_{n,n+1} \sin^2 \beta + u_{n,n+1} \cos^2 \beta \tag{18}$$

$$U_{xx} = U_{n,n+1} \cos^2 \beta + u_{n,n+1} \sin^2 \beta \tag{19}$$

$$u_{zx} = U_{xz} = (U_{n,n+1} - u_{n,n+1}) \sin \beta \cos \beta \tag{20}$$

$$g_{zz} = G_{n,n+1} \sin^2 \beta + g_{n,n+1} \cos^2 \beta \tag{21}$$

$$G_{xx} = G_{n,n+1} \cos^2 \beta + g_{n,n+1} \sin^2 \beta \tag{22}$$

$$g_{zx} = G_{xz} = (G_{n,n+1} - g_{n,n+1}) \sin \beta \cos \beta \tag{23}$$

Illustration of an element of the blade with the dynamic forces and moments acting on it along the flatwise and edgewise principal axes are given in Figures 9 and 11 respectively. Considering the forces acting on an element of the blade, related equations for flatwise, edgewise or coupling can be derived using the coupling terms above.

1. Myklestad Method Applied to Rotor Blade About Flatwise Axis

The rotor blade, which can be considered as a twisted beam, is divided into 'n' sections. As described in the beginning of this chapter, the mass of each section is assumed to be concentrated at the spanwise c.g of each section. Each section has the same properties as shown in Figure 9. A frequency is assumed, and the deflection curve is calculated. When the deflection curve satisfies the 'Boundary Conditions' then the assumed frequency is the uncoupled natural bending frequency of the blade (or beam), and the deflection is the normal bending mode (mode shape) of the blade.

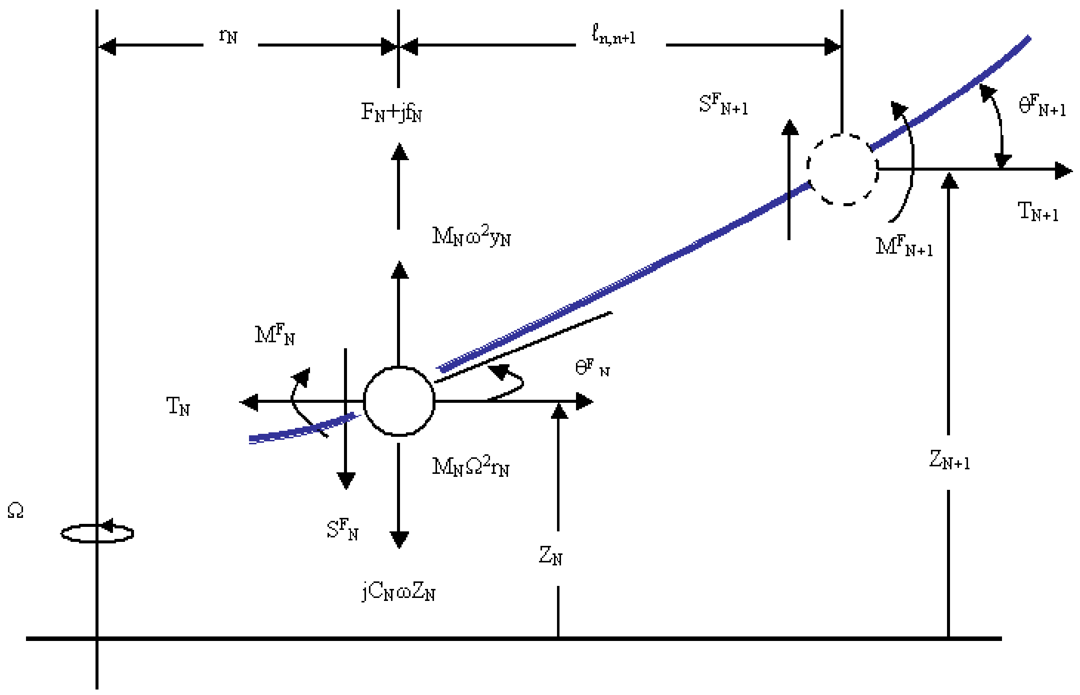


Figure 9. Blade Element Equilibrium; Flatwise System "After [Ref. 12]"

In this methodology, according to the Figure 9, we will describe from the tip to the root with 'n+1" closer to the tip and "n" closer to the root of the blade. If we consider the forces acting on the blade the "Equations of Motion" can be described as follows:

The assumptions for applying the method for an eigenvalue analysis are as follows:

- Flatwise Aerodynamic Damping constant is neglected;

$$C_n = 0$$

- Aerodynamic Lift Force acting on the blade elements for a particular frequency component, $F_N \sin\omega t + f_N \cos\omega t$, is neglected;

$$F_n + jf_n = 0$$

- Aerodynamic Drag Force acting on the blade elements for a particular frequency component, $D_n \sin\omega t + d_n \cos\omega t$, is neglected;

$$D_n + jd_n = 0$$

a. Centrifugal Force

Summing forces on this element

$$\sum F_x = 0 = -T_n + T_{n+1} + m\Omega^2 r_n$$

Then the "Centrifugal Tension" will be given by

$$T_n = T_{n+1} + m_n \Omega^2 r_n \tag{24}$$

b. Shear

Summing of these forces on this element

$$\sum F_z = 0 = -S_n + S_{n+1} + m_n \omega^2 Z_n - jC_n \omega Z_n + F_n + jf_n$$

Flatwise Shear can be written as:

$$S_n^F = S_{n+1}^F + m_n \omega^2 Z_n - jC_n \omega Z_n + F_n + jf_n \tag{25}$$

The term C_n represents the flatwise aerodynamic damping on the blade element, and can be expressed by

$$C_n = 5.73(chord)_n (l_{n,n+1})(\rho / 2)(\Omega r_n)$$

Thus after applying the assumptions listed previously Equation (25) becomes:

$$S_n^F = S_{n+1}^F + m_n \omega^2 Z_n$$

c. Moment

Real and imaginary flatwise equations for moment are the same. Writing the real equations, the corresponding equations are as follows:

$$M_n^F = M_{n+1}^F + S_{n+1}^F l_{n,n+1} - T_{n+1}(Z_{n+1} - Z_n) \tag{26}$$

d. Slope

Real and imaginary flatwise equations for slope are the same. Writing the real equations, the corresponding equations are as follows:

$$\begin{aligned}\theta^F_n = \;& \theta^F_{n+1}(1+T_{n+1}u_{zz}) + T_{n+1}\theta^E_{n+1}u_{zx} - M^F_{n+1}v_{zz} - M^E_{n+1}v_{zx} \\ & -S^F_{n+1}u_{zz} - S^E_{n+1}u_{zx}\end{aligned} \tag{27}$$

In the flatwise calculations the variables of edgewise principal axis (θ^E, M^E and S^E) are equal to 0, and the terms of coupling in Equation (27) are as follows: $u_{zz} = \frac{l^2_{n,n+1}}{2EI}$ (u_{zx} does not effect the equation because of being multiplied by zero value θ^E and S^E) and $v_{zz} = \frac{l_{n,n+1}}{EI}$ (v_{zx} is does not effect the equation because of being multiplied by zero value M^E). According to these assumptions the slope equation becomes:

$$\theta^F_n = \theta^F_{n+1}(1+T_{n+1}\frac{l^2_{n,n+1}}{2EI}) - M^F_{n+1}\frac{l^2_{n,n+1}}{EI} - S^F_{n+1}\frac{l^2_{n,n+1}}{2EI}$$

e. Deflection

Real and imaginary flatwise equations for deflection are the same. Writing the real equations, the corresponding equations become:

$$\begin{aligned}Z_n = \;& Z_{n+1} - \theta^F_n l_{n,n+1} + T_{n+1}\theta^F_{n+1}g_{zz} + T_{n+1}\theta^E_{n+1}g_{zx} \\ & -M^F_{n+1}u_{zz} - M^E_{n+1}u_{zx} - S^F_{n+1}g_{zz} - S^E_{n+1}g_{zx}\end{aligned} \tag{28}$$

In the calculations about the flatwise principal axis, (θ^E, M^E and S^E) are equal to 0, and the coefficients of coupling in Equation (28) are placed as $g_{zz} = \frac{l^3_{n,n+1}}{3EI}$ (g_{zx} does not affect the equation because of being multiplied by zero value θ^E and S^E)

and $u_{zx} = \frac{l^2_{n,n+1}}{2EI}$ (u_{zx} does not effect the equation because of being multiplied by zero value M^E). According to these assumptions the deflection equation becomes:

$$Z_n = Z_{n+1} - \theta_n l_{n,n+1} + T_{n+1}\theta_{n+1}\frac{l^3_{n,n+1}}{3EI} - M^F_{n+1}\frac{l^2_{n,n+1}}{2EI} - S^F_{n+1}\frac{l^3_{n,n+1}}{3EI}$$

To find the natural frequency of the rotor blade, the frequency, ω, is assumed, and shears, moments, slopes and deflections are calculated element by element from the tip of the blade to the root. At the tip of the blade, shears and moments will be equal to zero and slope and deflection of the blade will have some values:

$$S^F_T = 0$$
$$M^F_T = 0$$
$$\theta^F_T = \theta^F_T$$
$$Z_T = Z_T$$

Thus, there are two unknowns, Z_T and θ^F_T. These are carried along as unknowns, as we go element by element down the blade.

At the root of the blade we could write: (using subscript "$_0$" as the root)

$$S^F_0 = a_S Z_T + b_S \theta^F_T$$
$$M^F_0 = a_M Z_T + b_M \theta^F_T$$
$$\theta^F_0 = a_\theta Z_T + b_\theta \theta^F_T$$
$$Z_0 = a_y Z_T + b_y \theta^F_T$$

In these equations the constants "a" and "b" are generated as we go down the blade. They are functions of the mass and stiffness properties of the blade, the rotational speed (*Ω*), and the assumed frequency (*ω*).

At the root of the blade (for Hinged Blade):

$$M_0^F = 0$$

$$Z_0 = 0$$

Or,

$$a_M Z_T + b_M \theta_T^F = 0$$

$$a_z Z_T + b_z \theta_T^F = 0$$

In Matrix form:

$$\begin{bmatrix} a_M & b_M \\ a_z & b_z \end{bmatrix} \begin{bmatrix} Z_T \\ \theta_T \end{bmatrix} = 0$$

At the root of the blade (for Hingeless Blade):

$$\theta_0 = 0$$

$$Z_0 = 0$$

Or,

$$a_\theta Z_T + b_\theta \theta_T = 0$$

$$a_x Z_T + b_z \theta_T = 0$$

In Matrix form:

$$\begin{bmatrix} a_\theta & b_\theta \\ a_z & b_z \end{bmatrix} \begin{bmatrix} Z_T \\ \theta_T \end{bmatrix} = 0$$

Since the values of Z_T, and θ_T are non-zero, the solution to this equation is found when the determinant goes to zero. At this point the assumed natural frequency is the correct natural frequency. A graphical illustration of this procedure to find the natural frequency of the rotor blade is shown in Figure 10.

Once the blade natural frequencies have been established for various rotor speeds we can also construct a fanplot or "Southwell Plot". Since the blade is stiffer in the edgewise direction, the natural frequencies of the edgewise modes are higher than the flatwise modes.

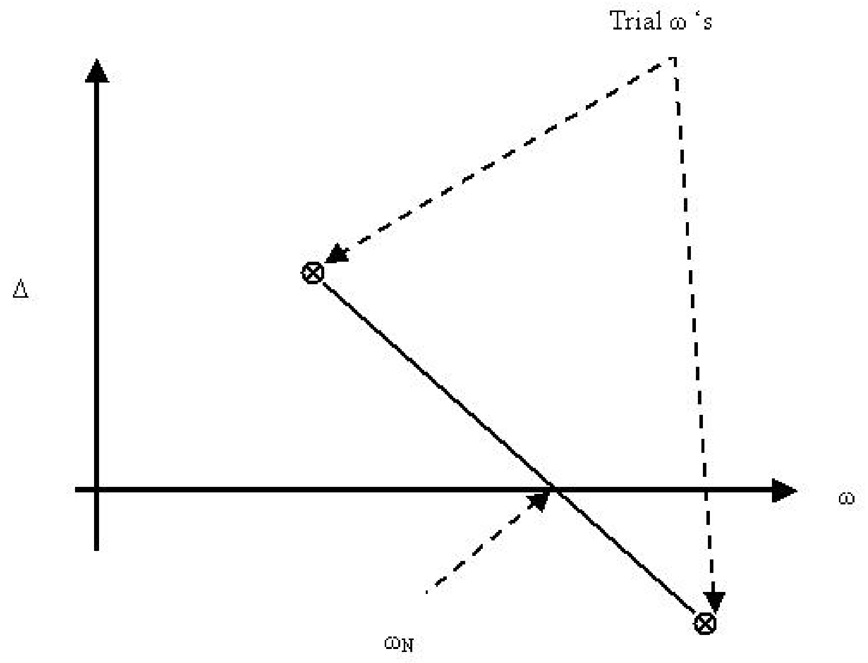


Figure 10. Determination of Natural Frequencies "From [Ref. 11]"

2. Myklestad Method Applied to Rotor Blade About Edgewise Axis

Using the same procedures that were applied to Flatwise Axis, the equations for Edgewise Axis can also be found. The forces acting on the blade element can be seen in Figure 11.

a. Centrifugal Force

Same as in Equation (24).

b. Shear

$$S_n^E = S_{n+1}^E + m_n(\omega^2 + \Omega^2)X_n + D_n + jd_n \tag{29}$$

After the assumptions are applied, Equation (29) becomes

$$S_n^E = S_{n+1}^E + m_n(\omega^2 + \Omega^2)X_n$$

c. Moment

$$M_n^E = M_{n+1}^E + S_{n+1}^E l_{n,n+1} - T_{n+1}(X_{n+1} - X_n) \tag{30}$$

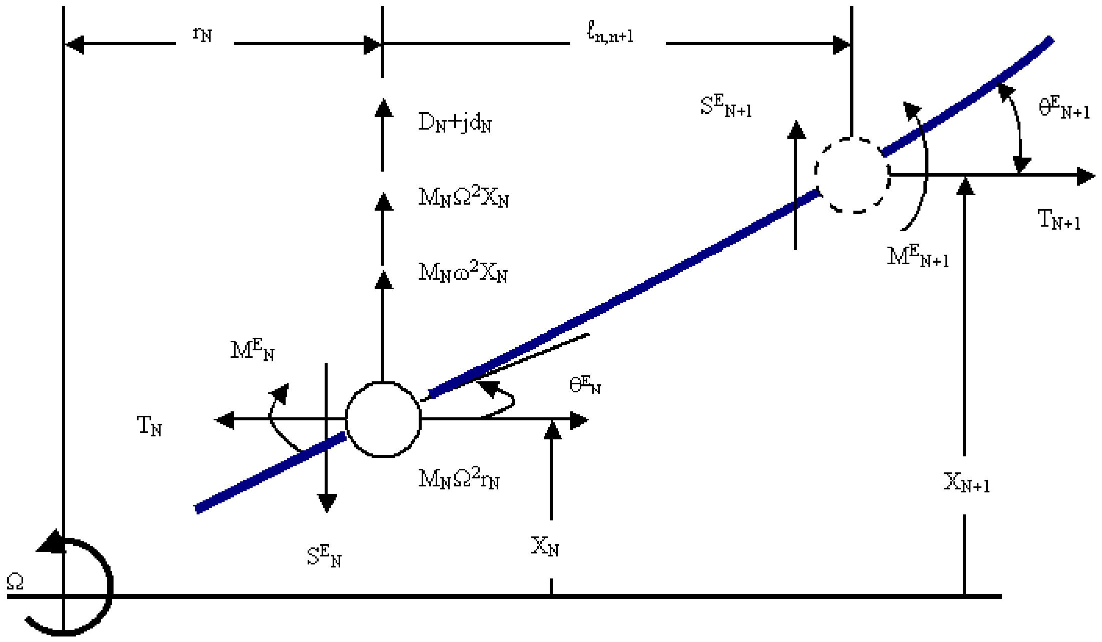


Figure 11. Blade Element Equilibrium; Edgewise System “After [Ref. 12]”

d. Slope

$$\begin{aligned}\theta_n^E = \; & \theta_{n+1}^E(1 + T_{n+1}U_{xx}) + T_{n+1}\theta_{n+1}^F U_{xz} - M_{n+1}^E V_{xx} - M_{n+1}^F V_{zx} \\ & - S_{n+1}^E U_{xx} - S_{n+1}^F U_{xz}\end{aligned} \tag{31}$$

In the edgewise calculations the variables of flatwise principal axis (θ^F, M^F and S^F) are equal to 0, and the coefficients of coupling in Equation (31) become: $U_{xx} = \frac{l^2_{n,n+1}}{2EI}$ (U_{xz} does not effect the equation because of being multiplied by zero value θ^F and S^F) and $V_{xx} = \frac{l_{n,n+1}}{EI}$ (V_{xz} is does not effect the equation because of being multiplied by zero value M^F). Accordingly, the slope equation becomes:

$$\theta_n^E = \theta^E_{n+1}(1 + T_{n+1}\frac{l^2_{n,n+1}}{2EI}) - M^E_{n+1}\frac{l^2_{n,n+1}}{EI} - S^E_{n+1}\frac{l^2_{n,n+1}}{2EI}$$

e. Deflection

$$\begin{aligned} X_n^E = \; & X^E_{n+1} - \theta^E_{n+1} l_{n,n+1} + T_{n+1}\theta^E_{n+1} G_{xx} + T_{n+1}\theta^F_{n+1} G_{xz} \\ & - M^E_{n+1} U_{xx} - M^F_{n+1} U_{xz} - S^E_{n+1} G_{xx} - S^F_{n+1} G_{xz} \end{aligned} \tag{32}$$

In the calculations about the edgewise principal axis, (θ^F, M^F and S^F) are equal to 0, and the coefficients of coupling in Equation (32) become: $G_{xx} = \frac{l^3_{n,n+1}}{3EI}$ (G_{xz} does not effect the equation because of being multiplied by zero value θ^F and S^F) and $U_{xx} = \frac{l^2_{n,n+1}}{2EI}$ (U_{xz} does not effect the equation because of being multiplied by zero value M^E). Accordingly the deflection equation becomes:

$$X_n = X_{n+1} - \theta^E_n l_{n,n+1} + T_{n+1}\theta^E_{n+1}\frac{l^3_{n,n+1}}{3EI} - M^E_{n+1}\frac{l^2_{n,n+1}}{2EI} - S^E_{n+1}\frac{l^3_{n,n+1}}{3EI}$$

Using the equations above along with the boundary conditions, which depend on the blade shape (hingeless or hinged), the same procedure can be followed as was in the flatwise axis, and the corresponding natural frequencies and mode shapes can be found.

3. Myklestad Method Applied to Rotor Blade About Coupled Axis

In the calculations for the coupled natural frequencies, the equations for both flatwise and edgewise principal axis are included. As a result of using both principal axes; matrix dimension expands from 4X4 matrix to 8X8 matrix; and the boundary condition matrix expands from 2X2 matrix to 4X4 matrix. Additionally, because of having blade element feathered at an angle of β, which includes twist β_T and blade feathering θ; the rotor blade twist coupling coefficients, should be included in all equations. Using the same procedure applied to both flatwise and edgewise axes, the following equations can be found.

a. Centrifugal Force

$$T_n = T_{n+1} + m_n \Omega^2 r_n$$

b. Shear

$$S_n^F = S_{n+1}^F + m_n \omega^2 Z_n - jC_n \omega Z_n + F_n + jf_n$$

$$S_n^E = S_{n+1}^E + m_n (\omega^2 + \Omega^2) X_n + D_n + jd_n$$

c. Moment

$$M_n^F = M_{n+1}^F + S_{n+1}^F l_{n,n+1} - T_{n+1}(Z_{n+1} - Z_n)$$

$$M_n^E = M_{n+1}^E + S_{n+1}^E l_{n,n+1} - T_{n+1}(X_{n+1} - X_n)$$

d. Slope

Now all the coefficients for twist coupling will be used as described in both flatwise and edgewise axes applications.

$$\theta_n^F = \theta_{n+1}^F(1+T_{n+1}u_{zz})+T_{n+1}\theta_{n+1}^E u_{zx} - M_{n+1}^F v_{zz} - M_{n+1}^E v_{zx} - S_{n+1}^F u_{zz} - S_{n+1}^E u_{zx}$$

$$\theta_n^E = \theta_{n+1}^E(1+T_{n+1}U_{xx})+T_{n+1}\theta_{n+1}^F U_{xz} - M_{n+1}^E V_{xx} - M_{n+1}^F V_{zx} - S_{n+1}^E U_{xx} - S_{n+1}^F U_{xz}$$

e. Deflection

Same as the slope, for deflection equations the coefficients of twist coupling are included in the equations.

$$Z_n = Z_{n+1} - \theta_n^F l_{n,n+1} + T_{n+1}\theta_{n+1}^F g_{zz} + T_{n+1}\theta_{n+1}^E g_{zx} - M_{n+1}^F u_{zz} - M_{n+1}^E u_{zx} - S_{n+1}^F g_{zz} - S_{n+1}^E g_{zx}$$

$$X_n^E = X_{n+1}^E - \theta_{n+1}^E l_{n,n+1} + T_{n+1}\theta_{n+1}^E G_{xx} + T_{n+1}\theta_{n+1}^F G_{xz} - M_{n+1}^E U_{xx} - M_{n+1}^F U_{xz} - S_{n+1}^E G_{xx} - S_{n+1}^F G_{xz}$$

By using the equations stated above and the boundary conditions, the natural frequencies of the blade can be determined. The natural frequencies are found in the same manner as in the previous sections. The only difference is that the dimension of the boundary conditions matrix is 4X4 instead of being 2X2.

III. BUILDING THE MATLAB® CODE AND VALIDATION OF THE CODE

A. APPLYING THE MATLAB® CODE TO MYKLESTAD ANALYSIS

The primary difference for calculations between Hinged (Articulated) and Hingeless (Rigid) Blade are the boundary conditions at the root. The boundary conditions at the tip are same for both Hinged (Articulated) and Hingeless (Rigid) Blades.

Centrifugal Force for all the calculations is the same as in Equation (24):

$$T_n = T_{n+1} + m_n \Omega^2 r_n$$

1. Matrix Form of the Hinged (Articulated) Blade

a. Flatwise

Following the application of Myklestad Method to the rotor blade about flatwise axis as it is explained in the previous chapter, the matrix form of the equations according to the boundary conditions is formed as follows:

The boundary conditions at the tip are

$$\begin{aligned} S^F{}_T &= 0 \\ M^F{}_T &= 0 \\ \theta^F{}_T &= \theta^F{}_T \\ Z_T &= Z_T \end{aligned} \quad \text{or} \quad X_{tip} = \begin{bmatrix} 0 \\ 0 \\ \theta^F{}_T \\ Z_T \end{bmatrix},$$

and the boundary conditions at the root for a hinged blade are

$$\begin{aligned} S^F{}_0 &= S^F{}_0 \\ M^F{}_0 &= 0 \\ \theta^F{}_0 &= \theta^F{}_0 \\ Z_0 &= 0 \end{aligned} \quad \text{or} \quad X_{root} = \begin{bmatrix} S_0 \\ 0 \\ \theta_0 \\ 0 \end{bmatrix}$$

If the equations proceed from tip to root for the flatwise axis, the equations can be written in the following matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & -m_n\omega^2 \\ 0 & 1 & 0 & -T_{n+1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & l_{n,n+1} & 1 \end{bmatrix} \begin{bmatrix} S^F{}_n \\ M^F{}_n \\ \theta^F{}_n \\ Z_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{n,n+1} & l & 0 & -T_{n+1} \\ -l^2/2EI & -l/EI & (1+-T_{n+1}l^2)/2EI & 0 \\ -l^3/3EI & -l^2/2EI & T_{n+1}l^3/3EI & 0 \end{bmatrix} \begin{bmatrix} S^F{}_{n+1} \\ M^F{}_{n+1} \\ \theta^F{}_{n+1} \\ Z_{n+1} \end{bmatrix}$$

which is in the form

$$[G]*\left[X_n\right]=[A]*\left[X_{n+1}\right]$$

Moving $[G]$ to the right side by pre-multiplying each side of the equation by $[G]^{-1}$ yields

$$\left[X_n\right]=[G]^{-1}*[A]*\left[X_{n+1}\right]$$

Letting $[F]=[G]^{-1}*[A]$, a simplified form of the equation becomes:

$$\left[X_n\right]=[F]*\left[X_{n+1}\right]$$

Moving along from tip to the root by saving the new values of the $[F]$ matrix through all stations, the relationship between the root and the tip of the blade can be written as:

$$\left[X_{root}\right]=\left[F^F\right]*\left[X_{Tip}\right]$$

After applying the boundary conditions at the root, new form of the equation becomes

$$\begin{bmatrix} S^F{}_0 \\ 0 \\ \theta^F{}_0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_S & b_S & c_S & d_S \\ a_M & b_M & c_M & d_M \\ a_\theta & b_\theta & c_\theta & d_\theta \\ a_z & b_z & c_z & d_z \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \theta^F{}_T \\ Z_T \end{bmatrix}$$

which can be simplified to a 2X2 matrix

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_M & d_M \\ c_Z & d_Z \end{bmatrix} * \begin{bmatrix} \theta^F{}_T \\ Y_T \end{bmatrix}$$

When the determinant of the 2X2 matrix above goes to zero we get the natural frequency value.

Once the natural frequency is found, corresponding relative deflection of the blade in every station can be found by placing the deflection at tip to a value of 1 ($Z_{Tip} = 1$). The deflection values in every station also give us the mode shape of the blade at that natural frequency. All the steps above are programmed in MATLAB® version 6.1[Ref. 16]. In the MATLAB® program, H-3 (S-61) helicopter data is used [Ref. 17]. To see the complete text of generated MATLAB® code, refer to APPENDIX A.

b. Edgewise

Using the same procedure as the flatwise case,the boundary conditions at the tip are

$$\begin{aligned} S^E{}_T &= 0 \\ M^E{}_T &= 0 \\ \theta^E{}_T &= \theta^E{}_T \\ X_T &= X_T \end{aligned} \quad \text{or} \quad X_{tip} = \begin{bmatrix} 0 \\ 0 \\ \theta^E{}_T \\ X_T \end{bmatrix}$$

while the boundary conditions at the root are

$$\begin{aligned} S^E{}_0 &= S^E{}_0 \\ M^E{}_0 &= 0 \\ \theta^E{}_0 &= \theta^E{}_0 \\ X_0 &= 0 \end{aligned} \quad \text{or} \quad X_{root} = \begin{bmatrix} S^E{}_0 \\ 0 \\ \theta^E{}_0 \\ 0 \end{bmatrix}$$

If the equations proceed from tip to root for the edgewise (chordwise) axis the equations can be written in the following matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & -m_n(\omega^2+\Omega^2) \\ 0 & 1 & 0 & -T_{n+1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & l_{n,n+1} & 1 \end{bmatrix} \begin{bmatrix} S^E{}_n \\ M^E{}_n \\ \theta^E{}_n \\ X_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{n,n+1} & l & 0 & -T_{n+1} \\ -l^2/2EI & -l/EI & (1+-T_{n+1}l^2)/2EI & 0 \\ -l^3/3EI & -l^2/2EI & T_{n+1}l^3/3EI & 0 \end{bmatrix} \begin{bmatrix} S^E{}_{n+1} \\ M^E{}_{n+1} \\ \theta^E{}_{n+1} \\ X_{n+1} \end{bmatrix}$$

which is in the form

$$[G]*\left[X_n\right]=[A]*\left[X_{n+1}\right]$$

Moving $[G]$ to the right side by pre-multiplying each side of the equation by $[G]^{-1}$ yields

$$\left[X_n\right]=[G]^{-1}*[A]*\left[X_{n+1}\right]$$

Letting $[F]=[G]^{-1}*[A]$ a simplified form of the equation becomes:

$$\left[X_n\right]=[F]*\left[X_{n+1}\right]$$

Moving along from tip to the root by saving the new values of the $[F]$ matrix through all stations, the relationship between the root and the tip of the blade can be written as:

$$\left[X_{root}\right]=\left[F^{E}\right]*\left[X_{Tip}\right]$$

After applying the boundary conditions at the root, new form of the equation becomes:

$$\begin{bmatrix} S^{E}{}_{0} \\ 0 \\ \theta^{E}{}_{0} \\ 0 \end{bmatrix} = \begin{bmatrix} a^{E}{}_{S} & b^{E}{}_{S} & c^{E}{}_{S} & d^{E}{}_{S} \\ a^{E}{}_{M} & b^{E}{}_{M} & c^{E}{}_{M} & d^{E}{}_{M} \\ a^{E}{}_{\theta} & b^{E}{}_{\theta} & c^{E}{}_{\theta} & d^{E}{}_{\theta} \\ a^{E}{}_{X} & b^{E}{}_{X} & c^{E}{}_{X} & d^{E}{}_{X} \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \theta^{E}{}_{T} \\ X_{T} \end{bmatrix}$$

which can also be simplified to a 2X2 matrix

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c^{E}{}_{\theta} & d^{E}{}_{\theta} \\ c^{E}{}_{X} & d^{E}{}_{X} \end{bmatrix} * \begin{bmatrix} \theta^{E}{}_{T} \\ X_{T} \end{bmatrix}$$

When the determinant of the 2X2 matrix above goes to zero, we get the natural frequency values for the edgewise modes.

c. Coupled

Combining both flatwise and edgewise equations as explained in the previous chapter and using the same procedure, the matrix can be formed as follows:

The boundary conditions at the tip are

$$\begin{aligned} &{S^F}_T = 0 \\ &{M^F}_T = 0 \\ &{\theta^F}_T = {\theta^E}_T \\ &Z_T = Z_T \\ &{S^E}_T = 0 \\ &{M^E}_T = 0 \\ &{\theta^E}_T = {\theta^E}_T \\ &X_T = X_T \end{aligned} \quad \text{or} \quad X_{tip} = \begin{bmatrix} 0 \\ 0 \\ {\theta^F}_T \\ Z_T \\ 0 \\ 0 \\ {\theta^E}_T \\ X_T \end{bmatrix}$$

and the boundary conditions at the root are

$$\begin{aligned} &{S^F}_0 = {S^F}_0 \\ &{M^F}_0 = 0 \\ &{\theta^F}_0 = {\theta^F}_0 \\ &{Z^F}_0 = 0 \\ &{S^E}_0 = {S^E}_0 \\ &{M^E}_0 = 0 \\ &{\theta^E}_0 = {\theta^E}_0 \\ &{Z^E}_0 = 0 \end{aligned} \quad \text{or} \quad X_{root} = \begin{bmatrix} {S^F}_0 \\ 0 \\ {\theta^F}_0 \\ 0 \\ {S^E}_0 \\ 0 \\ {\theta^E}_0 \\ 0 \end{bmatrix}$$

If the equations proceed from tip to root for coupling mode, the equations can be written in matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & -m_n\omega^2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -T_{n+1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & l_{n,n+1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -m_n(\omega^2+\Omega^2) \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -T_{n+1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{n,n+1} & 1 \end{bmatrix} \begin{bmatrix} S^F{}_n \\ M^F{}_n \\ \theta^F{}_n \\ Z_n \\ S^E{}_n \\ M^E{}_n \\ \theta^E{}_n \\ X_n \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l_{n,n+1} & 1 & 0 & -T_{n+1} & 0 & 0 & 0 & 0 \\ -u_{zz} & -v_{zz} & (1+T_{n+1}u_{zz}) & 0 & -u_{zx} & -v_{zx} & T_{n+1}u_{zx} & 0 \\ -g_{zz} & -u_{zz} & T_{n+1} & 1 & -g_{zx} & -u_{zx} & -T_{n+1}g_{zx} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & l_{n,n+1} & 1 & 0 & -T_{n+1} \\ -U_{xz} & -V_{xz} & T_{n+1}U_{xx} & 0 & -U_{xx} & -V_{xx} & (1+T_{n+1}U_{xx}) & 0 \\ -G_{xz} & -U_{xz} & T_{n+1}G_{xz} & 0 & -G_{xx} & -U_{xx} & T_{n+1}G_{xx} & 1 \end{bmatrix} \begin{bmatrix} S^F{}_{n+1} \\ M^F{}_{n+1} \\ \theta^F{}_{n+1} \\ Z_{n+1} \\ S^E{}_{n+1} \\ M^E{}_{n+1} \\ \theta^E{}_{n+1} \\ X_{n+1} \end{bmatrix}$$

which is also in the form

$$[G]*\left[X_n\right]=[A]*\left[X_{n+1}\right]$$

Pre-multiplying each side of the equation by $[G]^{-1}$ yields

$$\left[X_n\right]=[G]^{-1}*[A]*\left[X_{n+1}\right]$$

Again letting $\left[F\right]=\left[G\right]^{-1}*\left[A\right]$, simplified form of the equations becomes:

$$\left[X_n\right]=\left[F\right]*\left[X_{n+1}\right]$$

Moving along from tip to the root by saving the new values of the $\left[F\right]$ matrix through all stations, the relationship between the root and the tip of the blade can be written as:

$$\left[X_{root}\right]=\left[F_{Total}\right]*\left[X_{Tip}\right]$$

After applying the boundary conditions at the root, new form of the equation becomes

$$\begin{bmatrix} S^F{}_0 \\ 0 \\ \theta^F{}_0 \\ 0 \\ S^E{}_0 \\ 0 \\ \theta^E{}_0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_S & b_S & c_S & d_S & e_S & f_S & g_S & h_S \\ a_M & b_M & c_M & d_M & e_M & f_M & g_M & h_M \\ a_\theta & b_\theta & c_\theta & d_\theta & e_\theta & f_\theta & g_\theta & h_\theta \\ a_Z & b_Z & c_Z & d_Z & e_Z & f_Z & g_Z & h_Z \\ a_{S^E} & b_{S^E} & c_{S^E} & d_{S^E} & e_{S^E} & f_{S^E} & g_{S^E} & h_{S^E} \\ a_{M^E} & b_{M^E} & c_{M^E} & d_{M^E} & e_{M^E} & f_{M^E} & g_{M^E} & h_{M^E} \\ a_{\theta^E} & b_{\theta^E} & c_{\theta^E} & d_{\theta^E} & e_{\theta^E} & f_{\theta^E} & g_{\theta^E} & h_{\theta^E} \\ a_X & b_X & c_X & d_X & e_X & f_X & g_X & h_x \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \theta^F{}_T \\ Z_T \\ 0 \\ 0 \\ \theta^E{}_T \\ X_T \end{bmatrix}$$

which can be simplified to a 4X4 matrix

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_M & d_M & g_M & h_M \\ c_Z & d_Z & g_Z & h_Z \\ c_{M^E} & d_{M^E} & g_{M^E} & h_{M^E} \\ c_X & d_X & g_X & h_X \end{bmatrix} * \begin{bmatrix} \theta^F{}_T \\ Z_T \\ \theta^E{}_T \\ X_T \end{bmatrix}$$

When the determinant of the 4X4 matrix above goes to zero we get the natural frequency values.

2. Matrix Form of the Hingeless (Rigid) Blade

The only difference from the Hinged (Articulated) Blade is to change to boundary conditions at the tip. Other equations are as listed in the Hinged (Articulated) Blade section. For that reason the same equations are not repeated in this section.

a. Flatwise

The boundary conditions at the root are

$$\begin{aligned} S^F{}_0 &= S^F{}_0 \\ M^F{}_0 &= M^F{}_0 \\ \theta^F{}_0 &= 0 \\ Z_0 &= 0 \end{aligned} \quad \text{or} \quad X_{root} = \begin{bmatrix} S^F{}_0 \\ M^F{}_0 \\ 0 \\ 0 \end{bmatrix}$$

After applying the boundary conditions at the root, new form of the equation becomes

$$\begin{bmatrix} S^F{}_0 \\ M^F{}_0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_S & b_S & c_S & d_S \\ a_M & b_M & c_M & d_M \\ a_\theta & b_\theta & c_\theta & d_\theta \\ a_Z & b_Z & c_Z & d_Z \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \theta^F{}_T \\ Z_T \end{bmatrix}$$

which can be simplified to a 2X2 matrix

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_\theta & d_\theta \\ c_Z & d_Z \end{bmatrix} * \begin{bmatrix} \theta^F{}_T \\ Z_T \end{bmatrix}$$

Using the same procedure as described in first section of this chapter natural frequencies and mode shapes can be found.

b. Edgewise

The boundary conditions at the root are

$$\begin{aligned} S^E{}_0 &= S_0 \\ M^E{}_0 &= M^E{}_0 \\ \theta^E{}_0 &= 0 \\ X_0 &= 0 \end{aligned} \quad \text{or} \quad X_{root} = \begin{bmatrix} S^E{}_0 \\ M^E{}_0 \\ 0 \\ 0 \end{bmatrix}$$

After applying the boundary conditions at the root, new form of the equation becomes

$$\begin{bmatrix} S^E{}_0 \\ M^E{}_0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_{S^E} & b_{S^E} & c_{S^E} & d_{S^E} \\ a_{M^E} & b_{M^E} & c_{M^E} & d_{M^E} \\ a_{\theta^E} & b_{\theta^E} & c_{\theta^E} & d_{\theta^E} \\ a_X & b_X & c_X & d_X \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \theta^E{}_T \\ X_T \end{bmatrix}$$

which can be simplified to a 2X2 matrix

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_{M^E} & d_{M^E} \\ c_X & d_X \end{bmatrix} * \begin{bmatrix} \theta_T \\ X_T \end{bmatrix}$$

Using the same procedure as described in first section of this chapter natural frequencies and mode shapes can be found

c. Coupled

The boundary conditions at the root are

$$\begin{aligned} S^F{}_0 &= S^F{}_0 \\ M^F{}_0 &= M^F{}_0 \\ \theta^F{}_0 &= 0 \\ Z^F{}_0 &= 0 \\ S^E{}_0 &= S^E{}_0 \\ M^E{}_0 &= M^E{}_0 \\ \theta^E{}_0 &= 0 \\ Z^E{}_0 &= 0 \end{aligned} \qquad \text{or} \quad X_{root} = \begin{bmatrix} S^F{}_0 \\ M^F{}_0 \\ 0 \\ 0 \\ S^E{}_0 \\ M^E{}_0 \\ 0 \\ 0 \end{bmatrix}$$

After applying the boundary conditions at the root new form of the equation becomes

$$\begin{bmatrix} S^F{}_0 \\ M^F{}_0 \\ 0 \\ 0 \\ S^E{}_0 \\ M^E{}_0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_S & b_S & c_S & d_S & e_S & f_S & g_S & h_S \\ a_M & b_M & c_M & d_M & e_M & f_M & g_M & h_M \\ a_\theta & b_\theta & c_\theta & d_\theta & e_\theta & f_\theta & g_\theta & h_\theta \\ a_Z & b_Z & c_Z & d_Z & e_Z & f_Z & g_Z & h_Z \\ a_{S^E} & b_{S^E} & c_{S^E} & d_{S^E} & e_{S^E} & f_{S^E} & g_{S^E} & h_{S^E} \\ a_{M^E} & b_{M^E} & c_{M^E} & d_{M^E} & e_{M^E} & f_{M^E} & g_{M^E} & h_{M^E} \\ a_{\theta^E} & b_{\theta^E} & c_{\theta^E} & d_{\theta^E} & e_{\theta^E} & f_{\theta^E} & g_{\theta^E} & h_{\theta^E} \\ a_X & b_X & c_X & d_X & e_X & f_X & g_X & h_X \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ \theta^F{}_T \\ Z_T \\ 0 \\ 0 \\ \theta^E{}_T \\ X_T \end{bmatrix}$$

which can be simplified to a 4X4 matrix

$$\begin{bmatrix}0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}c_{\theta} & d_{\theta} & g_{\theta} & h_{\theta}\\ c_{z} & d_{z} & g_{z} & h_{z}\\ c_{\theta^E} & d_{\theta^E} & g_{\theta^E} & h_{\theta^E}\\ c_{x} & d_{x} & g_{x} & h_{x}\end{bmatrix}*\begin{bmatrix}\theta^F{}_T\\ Z_T\\ \theta^E{}_T\\ X_T\end{bmatrix}$$

When the determinant of the 4X4 matrix above goes to zero we get the natural frequency value, and using the same procedures above mode shapes can be found.

B. VALIDATION OF THE CODE

1. Comparison of the Uniform Nonrotating Hinged Blade (Supported-Free Beam) with Young and Felgar

Although the MATLAB® Programming was written in accordance with procedures described in each subsection above, a validation of the results must be made. A good validation method is to compare the results of the program with the uniform and nonrotating beam mode shapes values in Young and Felgar [Ref. 18]. Related data used for validation is included in APPENDIX B. The publication [Ref. 18] provides eigenvalues and eigenvectors for nonrotating uniform beams with a wide range of boundary conditions. But the MATLAB® code given in this thesis is generated for H-3 (S-61) helicopter, which has a rotating blade with nonuniform blade properties. For that reason in order to get accurate results, the program generated for H-3 (S-61) helicopter was modified as a uniform and nonrotating blade to compare with Young-Felgar data. The modified program is included in APPENDIX A. Only first three mode shapes were used for the validation, and the MATLAB® program was run for 10, 20, 30, 40, 50, 100, and 200 blade stations. The MATLAB® code matched with the values of tables given in Young and Felgar, the accuracy of the plots increasing as the number of the stations were increased as is expected.

Figure 12 shows the results of one of these comparisons. The dashed lines with green, red and magenta colors represent each mode shape of the corresponding Young and Felgar data, while the blue lines represent the mode shapes of the generated MATLAB® code. Tables 1 and 2 contain a summary of the comparisons.

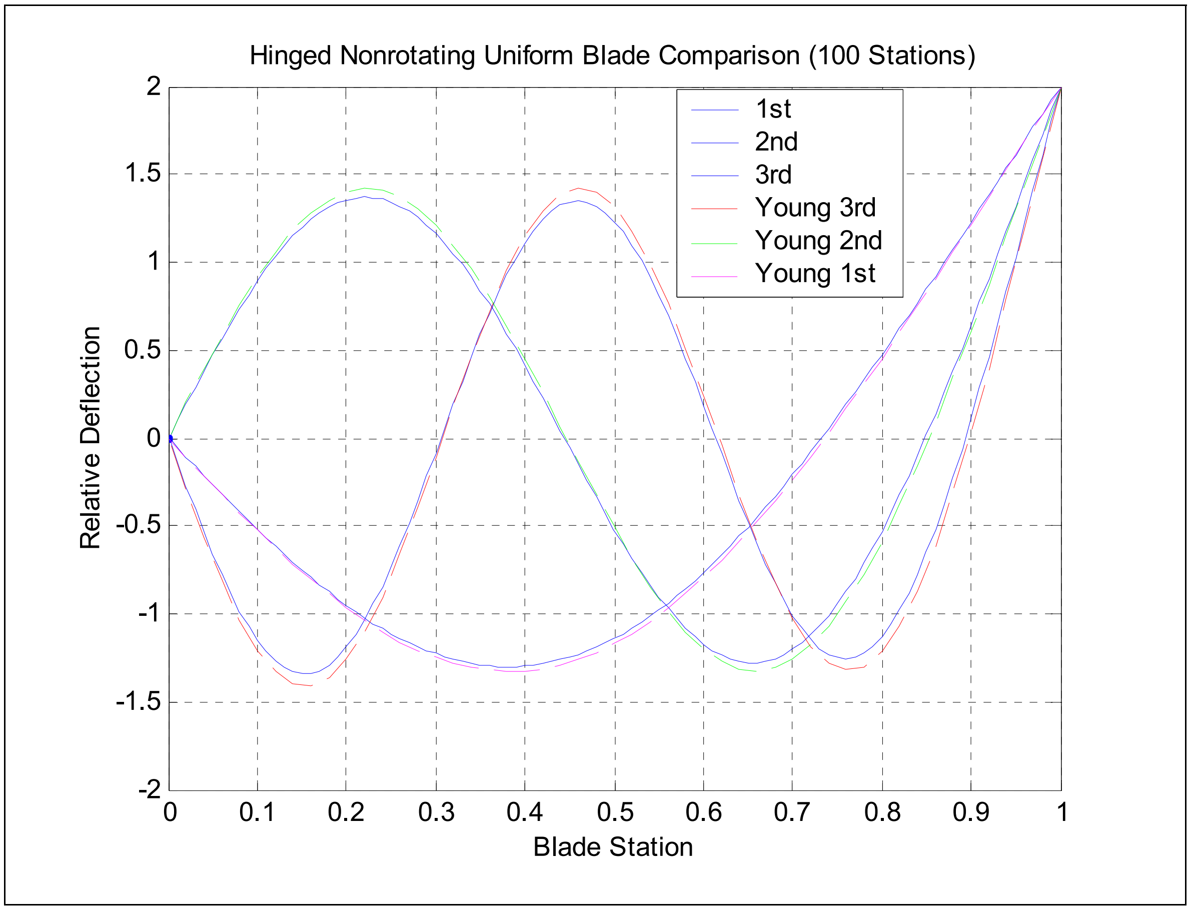


Figure 12. Comparison to Young and Felgar Table (Hinged Blade 100 Stations)

If we look through the results of all stations in Table 1 and Table 2 we can see that by increasing the number of stations the accuracy of the program increases as well.

	Young and Felgar	**10 Stations**	**20 Stations**	**30 Stations**	**40 Stations**	**50 Stations**	**100 Stations**	**200 Stations**
a_1	0	0	0	0	0	0	0	0
a_2	0.308580	0.306839	0.308158	0.308395	0.308477	0.308515	0.308564	0.308576
a_3	0.480431	0.476928	0.478647	0.479004	0.479129	0.479187	0.479264	0.479283

Table 1. Comparison of the Coefficients of Natural Frequencies

	10 Stations	20 Stations	30 Stations	40 Stations	50 Stations	100 Stations	200 Stations
a	0	0	0	0	0	0	0
a_2	-0.55644	-0.13699	-0.06001	-0.03344	-0.02132	-0.00522	-0.00131
a_3	-0.72907	-0.37131	-0.29696	-0.27088	-0.25884	-0.24287	-0.23891

Table 2. Error Ratios of the Natural Frequency Coefficients (%)

2. Comparison of the Uniform Nonrotating Hingeless Blade (Clamped-Free Beam) with Young and Felgar

Using a similar analysis as described in the previous section, Figure 13 shows the results of one of the comparisons for a uniform, nonrotating, hingeless blade.

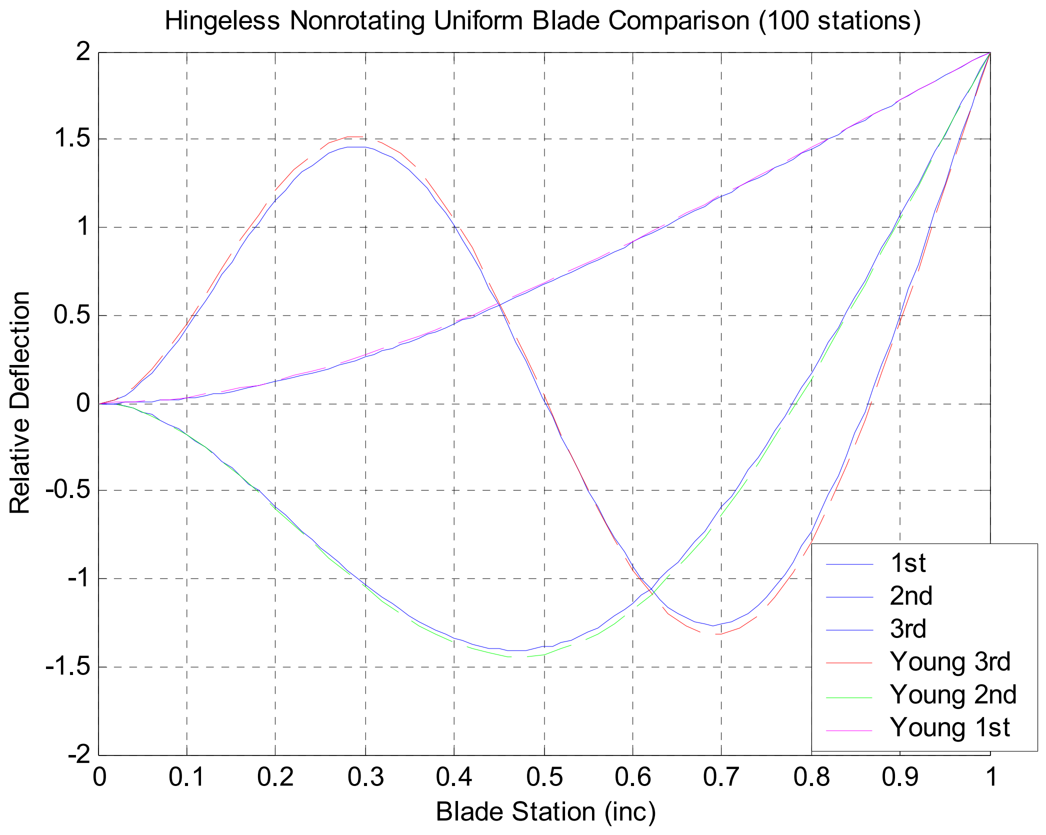


Figure 13. Comparison to Young and Felgar Table (Hingeless Blade 100 Stations)

If we look through the results of all stations in Table 3 and Table 4 it can be seen that increasing the number of stations improves the accuracy of the program as well.

	Young and Felgar	**10 Stations**	**20 Stations**	**30 Stations**	**40 Stations**	**50 Stations**	**100 Stations**	**200 Stations**
a_1	0.159523	0.148936	0.155566	0.155555	0.162790	0.162790	0.162790	0.162790
a_2	0.357139	0.353383	0.366000	0.365853	0.349593	0.355371	0.355371	0.356558

Table 3. Comparison of the Coefficients of Natural Frequencies

	10 Stations	**20 Stations**	**30 Stations**	**40 Stations**	**50 Stations**	**100 Stations**	**200 Stations**
a	-6.63674	-2.48727	-2.48726	2.048210	2.048210	2.048210	2.048210
a_2	-1.05161	2.45007	2.44008	-2.11281	-0.49484	-0.49484	-0.48251

Table 4. Error Ratios of the Natural Frequency Coefficients (%)

3. Analysis of H-3 Helicopter with an Articulated Rotor Blade

The first three mode shapes for H-3 (S-61) helicopter for both flatwise and edgewise axes are shown in Figures 14 and 15, respectively. Numerical values for the centrifugal force, natural frequencies, and the ratio of natural frequency to rotational speed are listed below each figure. Additional plots for the fourth and fifth mode shapes are attached in APPENDIX C.

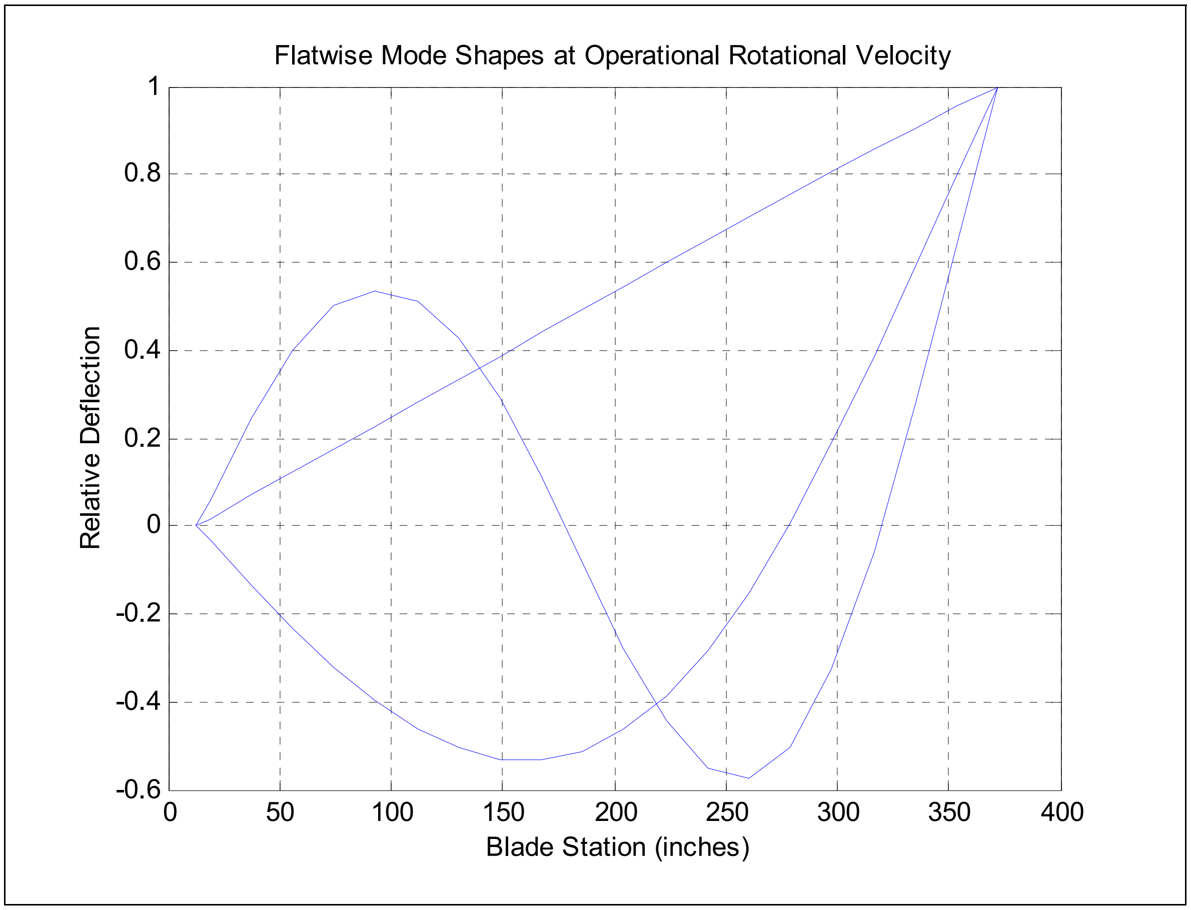


Figure 14. Flatwise Mode Shape for Hinged Blade

<u>The results for Flatwise Axis:</u>

Number of Mode shapes: 3

Centrifugal Force is : 4.515809216498805e+004

The Natural Frequencies are (cpm):

212.94221371078 554.01302042580 1006.88151545715

The ratio of the Natural Frequency to Rotational Speed is:

1.04897642222062 2.72912817943742 4.96000746530618

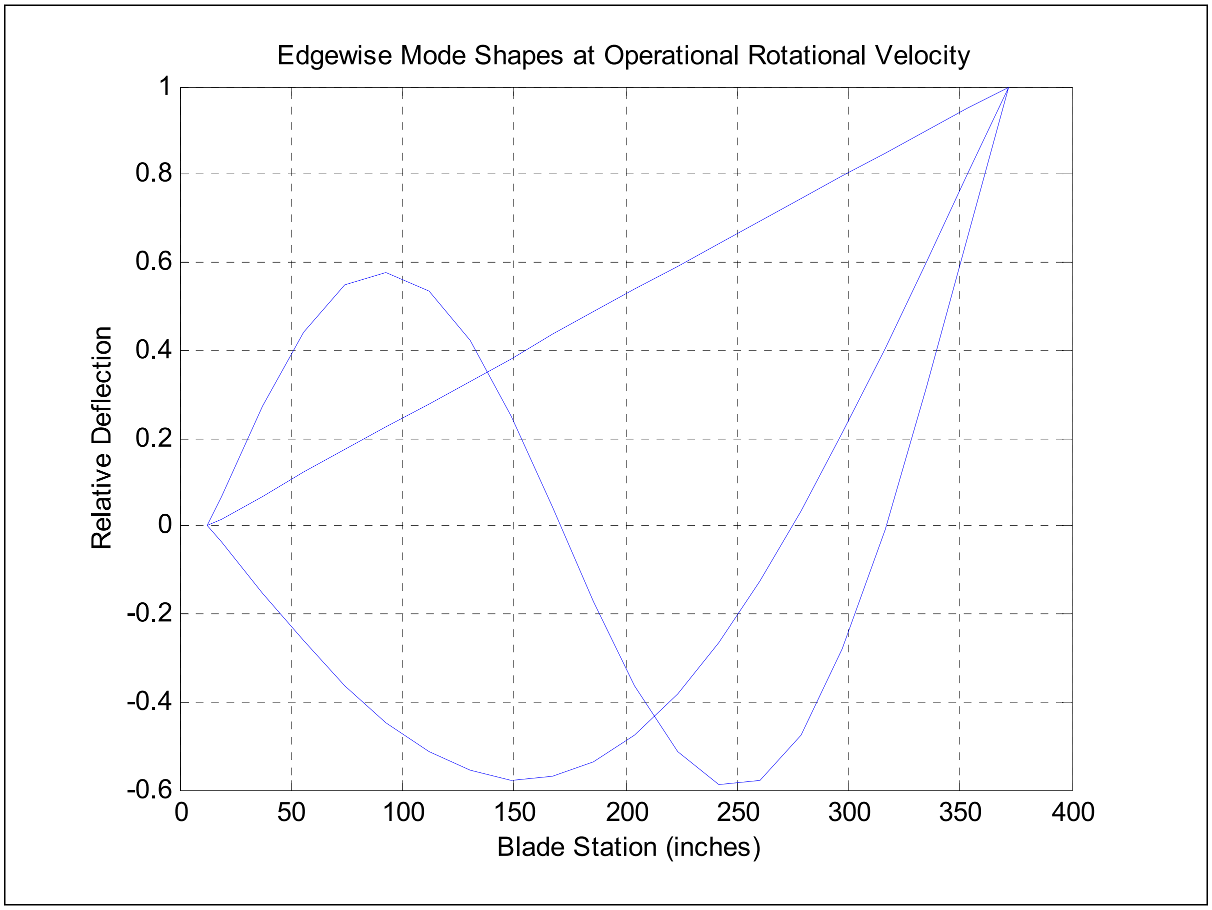


Figure 15. Edgewise Mode Shape for Hinged Blade

<u>The results for Edgewise Axis:</u>

Number of Mode shapes: 3

Centrifugal Force : 4.515809216498805 e+004

The Natural Frequencies are(cpm):

64.45769531250 713.63831103516 1828.18136248779

The ratio of the Natural Frequency to Rotational Speed is:

0.31752559267241 3.51545966027171 9.00581951964430

4. Analysis of H-3R Helicopter with a Hingeless Rotor Blade

The first three mode shapes for H-3R helicopter for both flatwise and edgewise axes are shown in Figures 16 and 17, respectively. Numerical values for the centrifugal force, natural frequencies, and the ratio of the natural frequency to rotational speed are listed below each figure. Additional plots for the fourth and fifth mode shapes are attached in APPENDIX C.

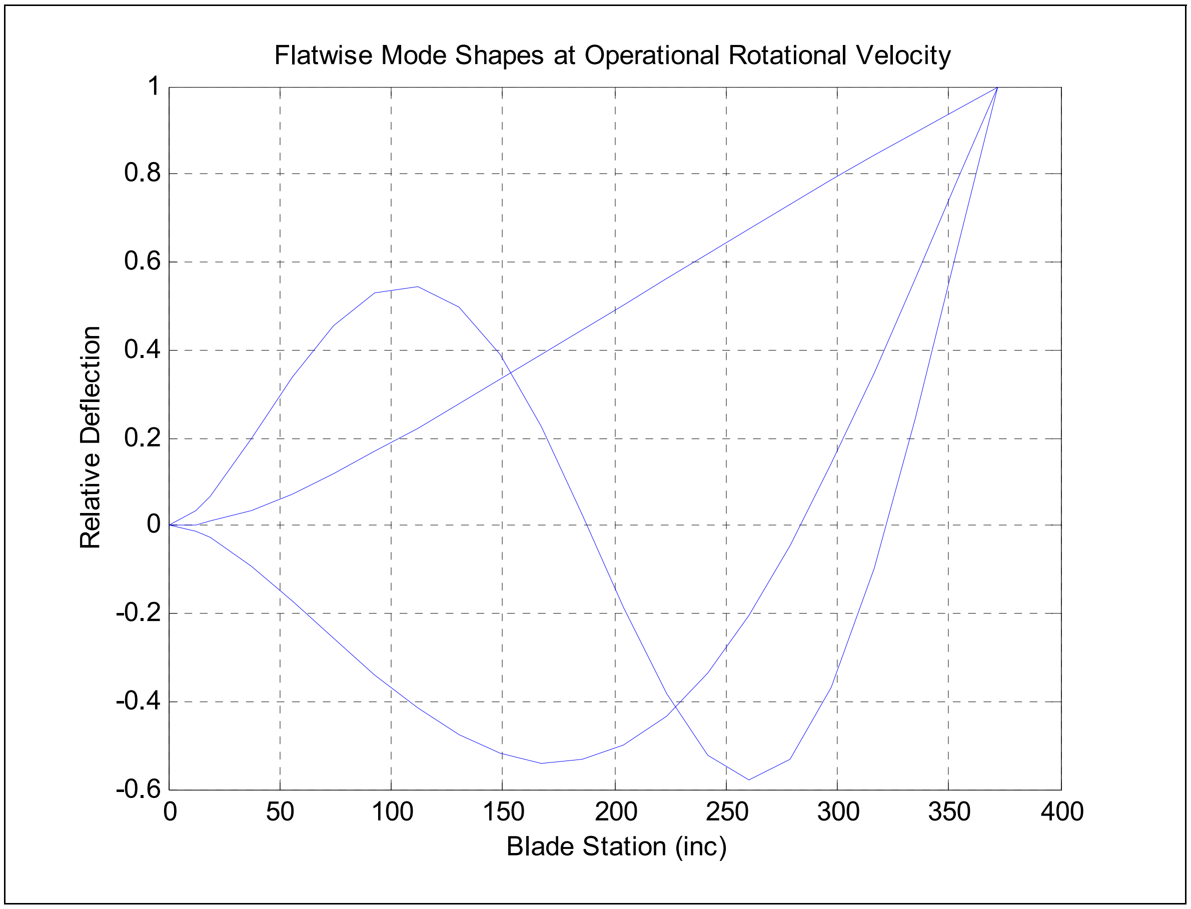


Figure 16. Flatwise Mode Shape for Hingeless Blade

<u>The results for Flatwise Axis:</u>

Number of Mode shapes: 3

Centrifugal Force : 4.515809216498805e+004

The Natural Frequencies are(cpm):

224.72100000000 582.81300000000 1068.79500000000

The ratio of the Natural Frequency/Rotational Speed is:

1.10700000000000 2.87100000000000 5.26500000000000

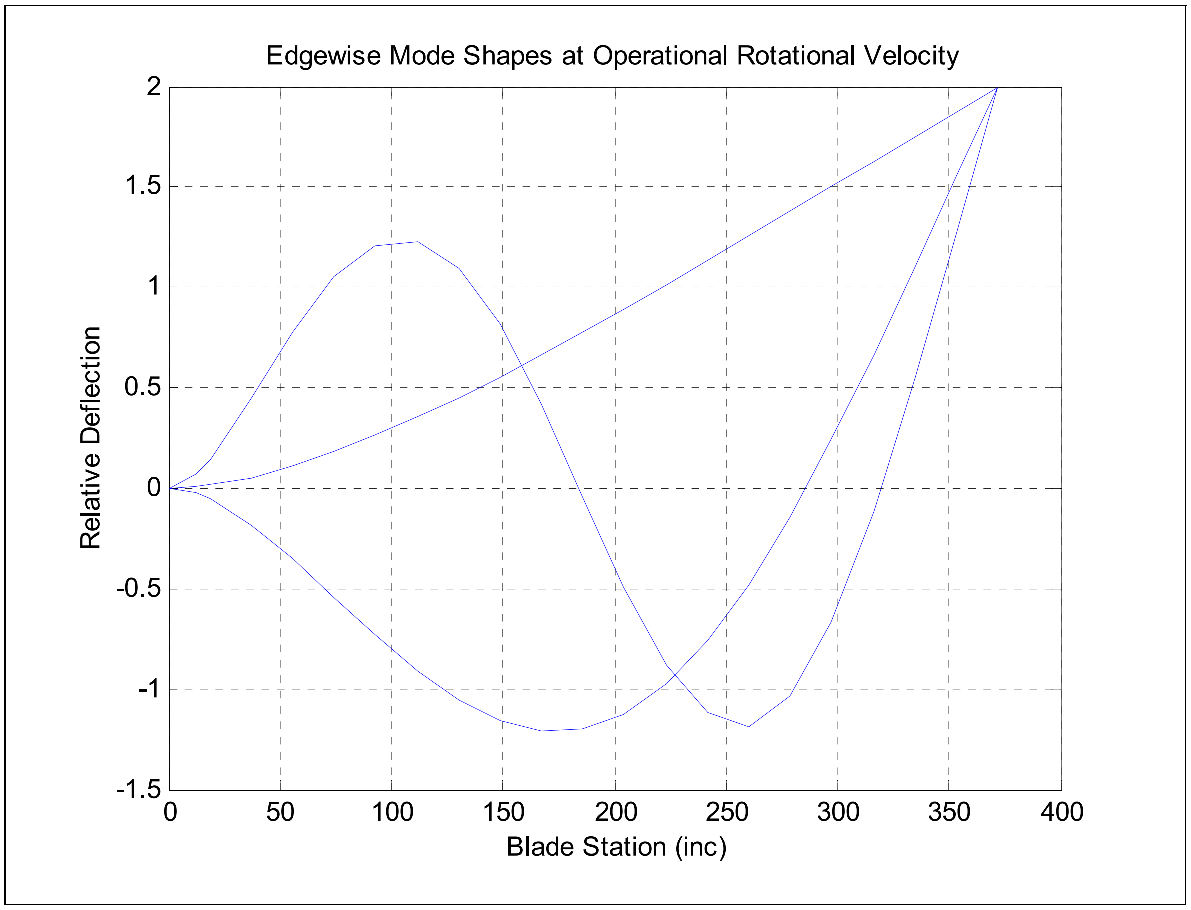


Figure 17. Edgewise Mode Shape for Hingeless Blade

<u>The results for Edgewise Axis:</u>

Number of Mode shapes: 3

Centrifugal Force : 4.515809216498805e+004

The Natural Frequencies are(cpm):

147.98700000000 842.24700000000 2051.72100000000

The ratio of the Natural Frequency/Rotational Speed is:

0.72900000000000 4.14900000000000 10.10700000000000

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS AND RECOMMENDATIONS

A useful method for ‘Blade Dynamic Analysis’ of a helicopter design is developed in this thesis. Myklestad Extension gives very good results when it is utilized with MATLAB®. A GUI is also developed in order to provide a user-friendly interface for the designer. A designer can utilize this program and find natural frequencies, mode shapes and centrifugal force for a given rotor blade with varying mass and stiffness distribution and any root and boundary conditions. By comparing the outputs with the actual data or recent research, the accuracy and the reliability of the program have been established and the GUI is verified.

In this thesis the theory and application of the theory to MATLAB® program is explained in detail. The full text of the MATLAB® codes is attached to APPENDIX A in order to have availability of further research on Myklestad Analysis and its application to JANRAD.

This thesis provides both uncoupled flatwise and edgewise analysis of the helicopter rotor blade. Although the theory of the coupled flatwise-edgewise case axis is presented in Chapter III, corresponding coupled MATLAB® codes have not been generated because of time. Assumptions for the analysis are given in Chapter II.

Blade dynamics is a complex area of research. Further research can be done on blade dynamics beyond the eigenvalue analysis presented here. If the blade forced response is desired, the assumptions of neglecting the damping, the aerodynamic lift force acting on the blade elements and the drag force acting on the blade elements can be removed and these values can be placed in the equations. Hence, a modified Myklestad Forced Response Analysis MATLAB® program can be generated. Also a coupled analysis (including flatwise-edgewise-torsion), which is not included in this thesis, can be added to the program for more detailed analysis on rotor blade dynamics.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB® PROGRAMS

A. HINGED BLADE ABOUT FLATWISE AXIS

```
clear
clc
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               MYKLESTAD ANALYSIS                                   %
%                                 BASIC PROGRAM                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TABLE B-1                                                                          %
%                                                                                    %
% BASIC BLADE DATA                                                                   %
%                                                                                    %
% H-1 HELICOPTER                                                                     %
%                                                                                    %
% 1-1 Planform, Articulated Blade                                                    %
% Gross Weight 12000lb. , 150Knot Design                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Radius of the stations from tip to root are(inch) : [372.00, 353.40, 334.80, 316.20,
%      297.60, 279.00, 260.40, 241.80, 223.20, 204.60, 186.00, 167.40, 148.80, 130.20,
%      111.60, 93.00, 74.40, 55.80, 37.20, 18.60, 12.63                              ];
%
R_n     = [372.00, 353.40, 334.80, 316.20, 297.60, 279.00, 260.40, 241.80, 223.20,...
           204.60, 186.00, 167.40, 148.80, 130.20, 111.60, 93.00 , 74.40 , 55.80 ,...
           37.20 , 18.60 , 12.63                                                     ];
% hinge(e) = 12.625 inches for H-1 Helicopter
%
%Flatwise Ixx(inch.^4) : [0.97, 1.93, 1.95, 1.99, 2.00, 2.04, 2.10, 2.19, 2.29, 2.35,
%       2.45, 2.51, 2.60, 2.71, 2.80, 2.91, 3.04, 4.40, 45.00, 75.00, 75.00         ];
%
I_xx    = [0.97, 1.93, 1.95, 1.99, 2.00, 2.04, 2.10, 2.19, 2.29, 2.35, 2.45, 2.51,...
           2.60, 2.71, 2.80, 2.91, 3.04, 4.40, 45.00, 75.00, 75.00                   ];
%
%Segment Weight(lb) : [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37,
%                      9.91, 9.94, 1.01, 9.73, 8.78, 9.06, 8.53, 10.51, 55.21, 84.17,
%                      20.46                                                         ];
```

```
%
W_n     = [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37, 9.91, 9.94,...
           10.01, 9.73, 8.78 , 9.06, 8.53, 10.51, 55.21, 84.17, 20.46               ];
%
m_n     = W_n/(32.174*12);
%
% Aliminum
E_n     = 1e7;
%
n       = length(R_n);
%
% n represents the number of the stations(segments);
%
%==========================================================================================
%
omega   = 0;
det_Bc  = 0;
%
modeshp = input('How many mode shapes would you like?  ');
%
R_V     = 203;                                  % RPM (R_V)
R_V1    = R_V * pi/30;                          % radians (R_V1)
%
%=========================================================================================
% Calculation of "Centrifugal Force" at the root:
%-----------------------------------------------------------------------------------------
T_n(1)  = m_n(1) * R_n(1) * R_V1^2;             % (T_n) and radians (R_V1)
%
% T_n(1) is the "Centrifugal Force at the TIP"
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * R_n(i+1) * R_V1^2;
end;                                        % end of loop for i--> { T_n(i) }
%
disp( ' Centrifugal Force is : ' )
disp(T_n(n))
% Displays "Centrifugal Force at r = 12.63"
%
%=========================================================================================
k       = 1;
```

```
i       = 1;
%
for omega_1 = 0 : modeshp^2*R_V/500 : 2.1*modeshp^2*R_V,
                                                % RPM (omega_1)
    omega(k) = omega_1*2*pi/60;                 % radians (omega(k))
    F = eye(4);
    %------------------------------------------------------------------------------------
    for j = 1:n-1,
        % determine length of the segments;
        l_sn = R_n(j) - R_n(j+1);
        %
        % detemine the stiffness
        EI = E_n * I_xx(j);
        %
        G_n = [ 1,     0,        0,          -m_n(j+1)*omega(k)^2;
                0,     1,        0,          -T_n(j)              ;
                0,     0,        1,           0                   ;
                0,     0,        l_sn,        1                  ];

        A_n = [ 1,                 0,                 0,                        0      ;
                l_sn,              1,                 0,                       -T_n(j) ;
               -(l_sn^2)/(2*EI),  -l_sn/EI,           1+(T_n(j)*l_sn^2)/(2*EI), 0      ;
               -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),   (T_n(j)*l_sn^3)/(3*EI),   1     ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                                        % end of loop for j --> { A_n, G_n, F_n )
    %
    %-------------------------------------------------------------------------------------
    B_c = [ F(2,3),F(2,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);
    %
    %-------------------------------------------------------------------------------------
    %determination of the points and the natural frequencies where det crosses the "0 line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
   else
       if k >1,                                 % if loop (1.a)
```

```
        s = k-1;
            if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
            omega_natural(i) = (omega(k) + omega(:,s))/2;
            i = i +1;
            end;                                 % end of if (1.b)
        end;                                     % end of if (1.a)
    end;                                         % end of elseif (1)
    %
    k = k+1;
end;                                             % end of loop for omega_1
%
%===============================================================================
%
% Sorting the "Natural Frequencies"
%
% Sort and convert the unit of natural frequency to 'cpm'
for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;                                             % end of loop for (m)
%
%================================================================================
%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % radians (omega_0) ,RPM (omega_n)
    %
    omega_new = 100;                        % omega_new i  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    %-----------------------------------------------------------------------------
    % Determination of odd or even mode shapes
    %
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end                                     % end of 'if loop R'
```

```
%
%-----------------------------------------------------------------------------------------
% Initial values
flag = 1;
flag_1 = 1;
det_bcold = 0;
det_bcnew = 0;
%
%----------------------------------------------------------------------------------------
%
while flag == 1,
    F = eye(4);
    if omega_0 < 0,
       omega_0 = 0;                     % if natural frequency is negative, make it zero
    end                                 % end of if loop omega
    %
    for j = 1:n-1,
        %
        l_sn = R_n(j) - R_n(j+1);
        %
        EI = E_n * I_xx(j);
        %
        G_n=[1,     0,        0,        -m_n(j+1)*omega_0^2     ;
             0,     1,        0,        -T_n(j)                 ;
             0,     0,        1,         0                      ;
             0,     0,        l_sn,      1                      ];
        %
        A_n=[1,                 0,                   0,                              0       ;
             l_sn,              1,                   0,                            -T_n(j)   ;
            -(l_sn^2)/(2*EI),  -l_sn/EI,             1+(T_n(j)*l_sn^2)/(2*EI),       0       ;
            -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),    (T_n(j)*l_sn^3)/(3*EI),         1       ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                                % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    B_c = [ F(2,3), F(2,4) ; F(4,3), F(4,4) ];
    det_bc = det(B_c);
    %-------------------------------------------------------------------------------------
    %
```

```
if (abs(det_bc)<1000 | abs(omega_old-omega_new)<1e-10),
                                    % if loop 1
    omega_n(count) = omega_0*30/pi; % radians (omega_0), RPM (omega_n)
    %
    if abs(omega_old-omega_new)<1e-10,                             % if loop 1.a
        disp('Warning!!  This value of omega may be in error.')
    end                                                            % end of 'if loop 1.a'
    %
    % change the initial values
    flag = 0;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %
else                                                               % else for if loop 1
    %
    if flag_1 == 1,                                                % if loop 1.b
        if det_bc > 0,                                             % if loop 1.b.i
            omega_new = omega_0 - (200*pi/30)*slope;               % radians(omega_new)
        else                                                      % else for if loop 1.b.i
            omega_new = omega_0 + (200*pi/30)*slope;
        end;                                                      % end of 'loop if 1.b.i'
        %
        % Rearrange the values
        %
        omega_old = omega_0;
        omega_0 = omega_new;
        det_bcold = det_bc;
        flag_1 = 2;
        %
    elseif flag_1 == 2,                                          % elseif for if loop 1.b
        if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,  % if loop 1.b.ii
            if det_bc > 0,                                         % if loop 1.b.ii.a
                omega_new = omega_0 - (200*pi/30)*slope;
            else                                                % else for if loop 1.b.ii.a
                omega_new = omega_0 + (200*pi/30)*slope;
            end;                                                 % end of 'if loop 1.b.ii.a
            %
            % Rearrange the values
            omega_old = omega_0;
```

```
                omega_0 = omega_new;
                det_bcold = det_bc;

            else                                                % else for if loop 1.b.ii
                omega_0 = (omega_new + omega_old)/2;
                det_bcnew = det_bc;
                flag_1 = 0;
            end                                                 % end of 'if loop 1.b.ii'
        else                                                      % else for if loop 1-b
            if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),% if loop 1.b.iii
                omega_new = omega_0;
                det_bcnew = det_bc;
            elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),
                                                          % elseif for if loop 1.b.ii
                omega_old = omega_0;
                det_bcold = det_bc;
            end;                                            % end of 'if loop 1.b.ii'
            %
            omega_0 = (omega_new + omega_old)/2;
            %
        end;                                                       % end of 'if loop 1.b'
        %
    end;                                                           % end of 'if loop 1'
    %
  end;                                                             % end of 'while loop'
 %
 theta_n(count) = -B_c(1,2) / B_c(1,1);
 % According to the equation we can also use as : theta_n(count)= -B_c(2,2)/B_c(2,1)
 %
end                                                              % end of loop for 'count'
%
%=================================================================================
%
for m= 1 : count                                                   % for loop 'm'
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                                 % for loop 'j'
        %
        l_sn    = R_n(j) - R_n(j+1);
        %
        EI      = E_n * I_xx(j);
        %
        G_n     =[ 1,     0,        0,      -m_n(j+1)*(omega_n(m)*pi/30)^2  ;
                   0,     1,        0,      -T_n(j)                         ;
                   0,     0,        1,       0                              ;
                   0,     0,      l_sn,             1                       ];
        %
        A_n     =[ 1,                0,                  0,                          0       ;
                   l_sn,             1,                  0,                        -T_n(j)  ;
                  -(l_sn^2)/(2*EI),  -l_sn/EI,           1+(T_n(j)*l_sn^2)/(2*EI),  0       ;
                  -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),  (T_n(j)*l_sn^3)/(3*EI),     1       ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
        %
        X_n(:,j+1) = F * X_tip;
        %
    end;                                  % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % Get the total deflection at each radial station
    %
    deflection(m,:)=X_n(4,:);
    plot ( R_n/R_n(1), 2*X_n(4,:), '-')
    grid on;
    title  ( ' Flapwise Mode Shapes at Operational Rotational Velocity ' );
    xlabel ( ' Blade Station (inches)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end                                                                        % end of for loop count
hold off
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency to Rotational Speed is: ')
disp(omega_n/R_V)
%
   % End of program for " HINGED-FLATWISE "
```

B. HINGED BLADE ABOUT EDGEWISE AXIS

```
clear
clc
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               MYKELSTAD ANALYSIS                                    %
%                                 BASIC PROGRAM                                       %
%                                 FOR EDGEWISE                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TABLE B-1                                                                           %
%                                                                                     %
% BASIC BLADE DATA                                                                    %
%                                                                                     %
% H-1 HELICOPTER                                                                      %
%                                                                                     %
% 1-1 Planform, Articulated Blade                                                     %
% Gross Weight 12000lb. , 150Knot Design                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Radius of the stations from tip to root are(inch) : [372.00, 353.40, 334.80, 316.20,
%      297.60, 279.00, 260.40, 241.80, 223.20, 204.60, 186.00, 167.40, 148.80, 130.20,
%      111.60, 93.00, 74.40, 55.80, 37.20, 18.60, 12.63                              ];
%
R_n     = [372.00, 353.40, 334.80, 316.20, 297.60, 279.00, 260.40, 241.80, 223.20,...
           204.60, 186.00, 167.40, 148.80, 130.20, 111.60, 93.00 , 74.40 , 55.80 ,...
           37.20 , 18.60 , 12.63                                                  ];
% hinge(e) = 12.625 inches for H-1 Helicopter
%
%
%Edgewise Ixx(inch.^4) :
%
I_yy    = [10.10, 20.30, 20.50, 20.60, 20.80, 20.90, 23.70, 24.00, 24.30, 24.80,...
           27.00, 27.30, 28.00, 28.50, 29.30, 29.80, 30.50, 35.00, 70.00, 75.00,...
           75.00
];
%
%Segment Weight(lb) : [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37,
%                      9.91, 9.94, 1.01, 9.73, 8.78, 9.06, 8.53, 10.51, 55.21, 84.17,
%                      20.46                                                         ];
%
W_n     = [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37, 9.91, 9.94,...
           10.01, 9.73, 8.78 , 9.06, 8.53, 10.51, 55.21, 84.17, 20.46             ];
%
m_n     = W_n/(32.174*12);
%
% Aliminum
E_n     = 1e7;
%
n       = length(R_n);
%
% n represents the number of the stations(segments);
%
%===============================================================================
==
%
omega   = 0;
det_Bc  = 0;
%
modeshp = input('How many mode shapes would you like?  ');
%
R_V     = 203;                                  % RPM (R_V)
R_V1    = R_V * pi/30;                          % radians (R_V1)
```

```
%
%===============================================================================
=
% Calculation of "Centrifugal Force" at the root:
%-------------------------------------------------------------------------------
-
T_n(1)  = m_n(1) * R_n(1) * R_V1^2;                  % (T_n) and radians (R_V1)
%
% T_n(1) is is the "Centrifugal Force at the TIP"
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * R_n(i+1) * R_V1^2;
end;                                             % end of loop for i--> { T_n(i) }
%
disp( ' Centrifugal Force : ' )
disp(T_n(n))
% Displays "Centrifugal Force at r = 12.625 "
%
%===============================================================================
=
k       = 1;
i       = 1;
%
for omega_1 = 0 : modeshp^2*R_V/500 : 2.1*modeshp^2*R_V,
                                                 % RPM (omega_1)
    omega(k) = omega_1*2*pi/60;                  % radians (omega(k))
    F = eye(4);
    %---------------------------------------------------------------------------
--
    for j = 1:n-1,
        % determine length of the segments;
        l_sn = R_n(j) - R_n(j+1);
        %
        % detemine the stiffness
        EI = E_n * I_yy(j);
        %
        G_n = [ 1,     0,        0,          -m_n(j+1)*(omega(k)^2 + R_V1^2);
                0,     1,        0,          -T_n(j)             ;
                0,     0,        1,           0                  ;
                0,     0,        l_sn,        1                 ];

        A_n = [ 1,                  0,                  0,                          0
;
                l_sn,             1,                  0,                        -T_n(j)
;
               -(l_sn^2)/(2*EI),   -l_sn/EI,               1+(T_n(j)*l_sn^2)/(2*EI),  0
;
               -(l_sn^3)/(3*EI),    -(l_sn^2)/(2*EI),      (T_n(j)*l_sn^3)/(3*EI),     1
];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                                        % end of loop for j --> { A_n, G_n,
F_n )
    %
    %---------------------------------------------------------------------------
----
    B_c = [ F(2,3),F(2,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);
    %
    %---------------------------------------------------------------------------
----
```

```
    %determination of the points and the natural frequencies where det crosses the "0
line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
   else
       if k >1,                                 % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                 % end of if (1.b)
       end;                                     % end of if (1.a)
   end;                                         % end of elseif (1)
   %
   k = k+1;
end;                                            % end of loop for omega_1
%
%=====================================================================================
=====
%
% Sorting the "Natural Frequencies"
%
% Sort and convert the unit of natural frequency to 'cpm'
for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;                                            % end of loop for (m)
%

%=====================================================================================
======
%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);       % radians (omega_0) ,RPM (omega_n)
    %
    omega_new = 100;                            % omega_new i  in RPM
    omega_old = -100;                           % omega_old is in RPM
    flg_slope = 1;
    %
    %---------------------------------------------------------------------------------
------
    % Determination of odd or even mode shapes
    %
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end                                         % end of 'if loop R'
    %
    %---------------------------------------------------------------------------------
------
    % Initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %
    %---------------------------------------------------------------------------------
-----
```

```
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
           omega_0 = 0;                    % if natural frequency is negative, make it
zero
        end                                    % end of if loop omega
        %
        for j = 1:n-1,
            %
            l_sn = R_n(j) - R_n(j+1);
            %
            EI = E_n * I_yy(j);
            %
            G_n=[1,     0,        0,        -m_n(j+1)*(omega_0^2 + R_V1^2)     ;
                 0,     1,        0,        -T_n(j)                     ;
                 0,     0,        1,         0                          ;
                 0,     0,        l_sn,      1                          ];
            %
            A_n=[1,             0,                  0,                           0
;
              l_sn,             1,                  0,                           -
T_n(j)  ;
              -(l_sn^2)/(2*EI),  -l_sn/EI,          1+(T_n(j)*l_sn^2)/(2*EI),        0
;
             -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),     (T_n(j)*l_sn^3)/(3*EI),         1
];
            %
            F_n = inv(G_n)*A_n;
            F = F_n * F;
        end;                                  % end of loo for j --> { l_sn, EI, A_n,
G_n, F_n)
        %
        B_c = [ F(2,3), F(2,4) ; F(4,3), F(4,4) ];
        det_bc = det(B_c);
        %----------------------------------------------------------------------------
-------
        %
        if (abs(det_bc)<1000 | abs(omega_old-omega_new)<1e-10),
                                               % if loop 1
            omega_n(count) = omega_0*30/pi;     % radians (omega_0), RPM (omega_n)
            %
            if abs(omega_old-omega_new)<1e-10,                              % if loop
1.a
                disp('Warning!!  This value of omega may be in error.')
            end                                                           % end of 'if
loop 1.a'
            %
            % change the initial values
            flag = 0;
            flag_1 = 1;
            det_bcold = 0;
            det_bcnew = 0;
            %
        else                                                             % else for if
loop 1
            %
            if flag_1 == 1,                                                 % if loop
1.b
                if det_bc > 0,                                              % if loop
1.b.i
                    omega_new  =  omega_0  -  (200*pi/30)*slope;                   %
radians(omega_new)
```

```
            else                                                     % else for if
loop 1.b.i
                omega_new = omega_0 + (200*pi/30)*slope;
            end;                                                 % end of 'loop if
1.b.i'
            %
            % Rearrange the values
            %
            omega_old = omega_0;
            omega_0 = omega_new;
            det_bcold = det_bc;
            flag_1 = 2;
            %
        elseif flag_1 == 2,                                    % elseif for if
loop 1.b
            if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,  % if loop
1.b.ii
                if det_bc > 0,                                       % if loop
1.b.ii.a
                    omega_new = omega_0 - (200*pi/30)*slope;
                else                                             % else for if loop
1.b.ii.a
                    omega_new = omega_0 + (200*pi/30)*slope;
                end;                                             % end of 'if loop
1.b.ii.a
                %
                % Rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;

            else                                                 % else for if loop
1.b.ii
                omega_0 = (omega_new + omega_old)/2;
                det_bcnew = det_bc;
                flag_1 = 0;
            end                                                  % end of 'if loop
1.b.ii'
        else                                                     % else for if
loop 1-b
            if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
% if loop 1.b.iii
                omega_new = omega_0;
                det_bcnew = det_bc;
            elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),

% elseif for if loop 1.b.ii

                omega_old = omega_0;

                det_bcold = det_bc;

            end;                                               % end of 'if loop 1.b.ii'

            %

            omega_0 = (omega_new + omega_old)/2;

            %

        end;                                                     % end of 'if loop 1.b'

        %

    end;                                                         % end of 'if loop 1'

    %
    end;                                                            % end of 'while loop'
 %
 theta_n(count) = -B_c(1,2) / B_c(1,1);
 % According to the equation we can also use as : theta_n(count)= -B_c(2,2)/B_c(2,1)
 %
end                                                                 % end of loop for 'count'
%
%===============================================================================================
%
for m= 1 : count                                                    % for loop 'm'
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                                  % for loop 'j'
        %
        l_sn    = R_n(j) - R_n(j+1);
        %
        EI      = E_n * I_yy(j);
        %
        G_n     =[ 1,     0,        0,      -m_n(j+1)*((omega_n(m)*pi/30)^2  + R_V1^2);
                   0,     1,        0,      -T_n(j)                        ;
                   0,     0,        1,       0                             ;
                   0,     0,      l_sn,            1                       ];
        %
        A_n     =[ 1,                 0,                 0,                          0       ;
                   l_sn,              1,                 0,                        -T_n(j)   ;
                  -(l_sn^2)/(2*EI),  -l_sn/EI,           1+(T_n(j)*l_sn^2)/(2*EI),  0       ;
                   -(l_sn^3)/(3*EI),   -(l_sn^2)/(2*EI),    (T_n(j)*l_sn^3)/(3*EI),         1
];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
        %
        X_n(:,j+1) = F * X_tip;
        %
    end;                               % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % Get the total deflection at each radial station
    %
    deflection(m,:)=X_n(4,:);
```

```
      plot ( R_n/R_n(1), 2*X_n(4,:), '-')
      grid on;
      title  ( ' Chordwise Mode Shapes at Operational Rotational Velocity ' );
      xlabel ( ' Blade Station (inches)' ) ;
      ylabel ( ' Relative Deflection ' ) ;
      hold on
  end                                                                % end of for loop count
  hold off
  disp('The Natural Frequencies are(cpm): ')
  disp(omega_n)
  %
  disp('The ratio of the Natural Frequency to Rotational Speed is: ')
  disp(omega_n/R_V)
  %
  % End of Program " Hinged-Edgewise "
```

C. HINGELESS BLADE ABOUT FLATWISE AXIS

```
clear
clc
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                       %
%                                     MYKELSTAD                                         %
%                               HINGELESS BLADE FLATWISE                                %
%                                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                       %
% Table B-17                                                                            %
%                                                                                       %
% Basic Blade Data                                                                      %
%                                                                                       %
% H1-R Helicopter                                                                       %
%                                                                                       %
% 1-1 Planform,Rigid Blade                                                              %
%                                                                                       %
% Gross Weight 12000lb. , 150 Knot Design                                               %
%                                                                                       %
%****************************************************************************************
%
% R_n = input('Enter the radius of the stations: ');
% Radius of the stations are( inch) : [372.00, 353.40, 334.80, 316.20, 297.60, 279.00,
% 260.40, 241.80, 223.20, 204.60, 186.00, 167.40, 148.80, 130.20, 111.60, 93.00, 74.40,
% 55.80, 37.20, 18.60, 12.63 ];
%
R_n     = [372.00, 353.40, 334.80, 316.20, 297.60, 279.00, 260.40, 241.80, 223.20,...
           204.60, 186.00, 167.40, 148.80, 130.20, 111.60,  93.00,  74.40,  55.80,...
           37.20, 18.60, 12.63,0                                                 ];
%
% Flatwise Ixx(inch.4)
%
I_xx    = [0.97, 1.93, 1.95, 1.99, 2.00, 2.04, 2.10, 2.19, 2.29, 2.35, 2.45, 2.51,...
           2.60, 2.71, 2.80, 2.91, 3.04, 4.40, 5.00, 5.00, 5.00                  ];
%
% Aliminum
E_n = 1e7;
%
n = length(R_n);
%
% Segment Weight(lb) : [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37, %
%9.91,
% 9.94, 1.01, 9.73, 8.78, 9.06, 8.53, 10.51, 55.21, 84.17, 20.46];
%
Weight_n = [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37, 9.91, 9.94,...
            10.01, 9.73, 8.78, 9.06, 8.53, 10.51, 55.21, 84.17, 20.46, 0         ];
%
m_n = Weight_n/(32.174*12);
%
omega = 0;
det_Bc = 0;
%
modeshp =input('How many mode shapes would you like?  ');
%
%***************************************************************************************
**
%
R_V = 203;                                                           % R_V is in RPM
R_V1= R_V * pi/30;                                                   % R_V1 is in radians
T_n(1) = m_n(1) * R_n(1) * R_V1^2;                                  % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * R_n(i+1) * R_V1^2;
```

```
end; % end of loop for i--> { T_n(i) }
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%*********************************************************************************
k = 1;
i = 1;
for omega1 = 0 : modeshp^2*R_V/500 : 2.1*modeshp^2*R_V,             % omega1 is in RPM
    omega(k) = omega1*2*pi/60;                                  % omega(k) is in radians
    F = eye(4);
    %----------------------------------------------------------------
    for j = 1:n-1,
        l_sn = R_n(j) - R_n(j+1);
        %
        EI = E_n * I_xx(j);
        %
        G_n=[1,     0,        0,          -m_n(j+1)*omega(k)^2; % omega(k) is in radians
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];
        %
        %
        A_n=[1,                0,                 0,                  0;
             l_sn,             1,                 0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                               % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %------------------------------------------------------------------
    B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);
    %
    if k >1,
       s = k-1;
       if det_bc(k) * det_bc(:,s) < 0
       omega_natural(i) = (omega(k) + omega(:,s))/2;
       i = i +1;
```

```
   end;
   end;
 %
k = k+1;
end;                                        % end of loop for omega1
%
%----------------------------------------------------------------------
%
for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;
%
%----------------------------------------------------------------------
%*************************************************************************************
%
for count = 1 : modeshp
    omega_0 = omega_n(count)*pi/30;          % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                                       % omega_new is  in RPM
    omega_old = -100;                                      % omega_old is in RPM
    flg_slope = 1;
    %
    %------------------------------------------------------------------
    % determination of odd or even mode shapes
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end                                                    % end of 'if loop R'
    %
    %-------------------------------------------------------------------
    %
    % initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %
    %-------------------------------------------------------------------
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
            omega_0 = 0;                 % if real part of natural frequency is negative
value make it zero
        end                             % end of if loop omega
        %
        %--------------------------------------------------------------
        for j = 1:n-1,
            %
            l_sn = R_n(j) - R_n(j+1);
            %
            EI = E_n * I_xx(j);
            %
            G_n=[1,     0,        0,        -m_n(j+1)*omega_0^2   ;            % omega_0
is in radians
                 0,     1,        0,        -T_n(j)           ;
                 0,     0,        1,         0                  ;
                 0,     0,        l_sn,      1                  ];
            %
            A_n=[1,                 0,                  0,                        0       ;
                 l_sn,              1,                  0,                       -T_n(j);
               -(l_sn^2)/(2*EI),  -l_sn/EI,      1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
               -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),  (T_n(j)*l_sn^3)/(3*EI)   1       ];
            %
            F_n = inv(G_n)*A_n;
            F = F_n * F;
        end;                          % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
        %
        B_c = [ F(3,3), F(3,4) ; F(4,3), F(4,4) ];
        det_bc = det(B_c);
        %
        %-------------------------------------------------------------
        if (abs(det_bc)<10000 | abs(omega_old-omega_new)<1e-10),            % if loop 1
            omega_n(count) = omega_0*30/pi;   % omega_0 is in radians, omega_n is in RPM
            %
            %----------------------------------------------------------
            %
            if abs(omega_old-omega_new)<1e-10,                          % if loop 1-a
                disp('Warning!!  This value of omega may be in error.')
            end                                                 % end of 'if loop 1-a'
            %
            %----------------------------------------------------------
            % change the initial values
            flag = 0;
            flag_1 = 1;
            det_bcold = 0;
            det_bcnew = 0;
            %
        else                                                        % else for if loop 1
            %---------------------------------------------------------
            if flag_1 == 1,                                               % if loop 1-b
                if det_bc > 0,                                            % if loop 1-b-i
                    omega_new = omega_0 - (200*pi/30)*slope;   % omega_new is in radians
                else                                           % else for if loop 1-b-i
                    omega_new = omega_0 + (200*pi/30)*slope;   % omega_new is in radians
                end;                                         % end of 'loop if 1-b-i'
                %-----------------------------------------------------
                % rearrange the values
                %
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                flag_1 = 2;
                %
            elseif flag_1 == 2,                                 % elseif for if loop 1-b
                if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                                            % if loop 1-b-ii
                    if det_bc > 0,                                     % if loop 1-b-ii-a
                        omega_new = omega_0 - (200*pi/30)*slope;%omega_new is in radians
                    else                                      % else for if loop 1-b-ii-a
                        omega_new = omega_0 + (200*pi/30)*slope;%omega_new is in radians
                    end;                                        % end of 'if loop 1-b-ii-a
                    %
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                else                                            % else for if loop 1-b-ii
```

```
                omega_0 = (omega_new + omega_old)/2;
                                         %omega_old,new.and 0 are in radians
                det_bcnew = det_bc;
                flag_1 = 0;
            end                                          % end of 'if loop 1-b-ii'
        else                                                 % else for if loop 1-b
            if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
                                                    % if loop 1-b-iii
                omega_new = omega_0;
                det_bcnew = det_bc;
            elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),
                                         % elseif for if loop 1-b-ii
                omega_old = omega_0;
                det_bcold = det_bc;
            end;                                         % end of 'if loop 1-b-ii'
            omega_0 = (omega_new + omega_old)/2;
        end;                                               % end of 'if loop 1-b'
        %
        %--------------------------------------------------------------------------
    end;                                                   % end of 'if loop 1'
    %------------------------------------------------------------------------------
  end;                                                       % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
 %
end % end of loop for count
%
*************************************************************************************
%
for m= 1 : count                                               % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                                     % for loop j
        %
        l_sn = R_n(j) - R_n(j+1);
        %
        EI = E_n * I_xx(j);
        %
        G_n=[1,     0,        0,   -m_n(j+1)*(omega_n(m)*pi/30)^2;
```

```
            0,     1,        0,         -T_n(j);
            0,     0,        1,              0;
            0,     0,       l_sn,             1];
        %
        %
        A_n=[1,                0,                 0,                 0;
             l_sn,             1,                 0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
        X_n(:,j+1) = F * X_tip;
    end;                                                    % end of loop for j -->
{ l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    figure(2)
    plot ( R_n, deflection(m,:), '-');
    grid on;
    title  ( ' Flapwise Mode Shapes at Operational Rotational Velocity ' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
%
%*****************************************************************************************
hold off
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency/Rotational Speed is: ')
disp(omega_n/R_V)
%
% End of the Program " Hingeless-Flatwise "
```

## D.	HINGELESS BLADE ABOUT EDGEWISE AXIS

```
clear
clc
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               MYKELSTAD ANALYSIS                                  %
%                                 BASIC PROGRAM                                     %
%                                 FOR EDGEWISE                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TABLE B-1                                                                         %
%                                                                                   %
% BASIC BLADE DATA                                                                  %
%                                                                                   %
% H-1 HELICOPTER                                                                    %
%                                                                                   %
% 1-1 Planform, Articulated Blade                                                   %
% Gross Weight 12000lb. , 150Knot Design                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Radius of the stations from tip to root are(inch) : [372.00, 353.40, 334.80, 316.20,
%      297.60, 279.00, 260.40, 241.80, 223.20, 204.60, 186.00, 167.40, 148.80, 130.20,
%      111.60, 93.00, 74.40, 55.80, 37.20, 18.60, 12.63                             ];
%
R_n     = [372.00, 353.40, 334.80, 316.20, 297.60, 279.00, 260.40, 241.80, 223.20,...
           204.60, 186.00, 167.40, 148.80, 130.20, 111.60, 93.00 , 74.40 , 55.80 ,...
           37.20 , 18.60 , 12.63 ,0                                                   ];
% hinge(e) = 12.625 inches for H-1 Helicopter
%
%
%Edgewise Ixx(inch.^4) :
%
I_yy    = [10.10, 20.30, 20.50, 20.60, 20.80, 20.90, 23.70, 24.00, 24.30, 24.80,...
           27.00, 27.30, 28.00, 28.50, 29.30, 29.80, 30.50, 35.00, 26.00, 26.00,...
           26.00                                                                     ];
%
%Segment Weight(lb) : [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37,
%                       9.91, 9.94, 1.01, 9.73, 8.78, 9.06, 8.53, 10.51, 55.21, 84.17,
%                       20.46                                                        ];
%
```

```
W_n      = [2.56, 12.57, 9.51, 9.19, 9.18, 9.29, 9.03, 9.14, 9.45, 9.37, 9.91, 9.94,...
           10.01, 9.73, 8.78 , 9.06, 8.53, 10.51, 55.21, 84.17, 20.46 ,0             ];
%
m_n      = W_n/(32.174*12);
%
% Aliminum
E_n      = 1e7;
%
n        = length(R_n);
%
% n represents the number of the stations(segments);
%
%==========================================================================================
%
omega    = 0;
det_Bc   = 0;
%
modeshp = input('How many mode shapes would you like?  ');
%
R_V      = 203;                                      % RPM (R_V)
R_V1     = R_V * pi/30;                              % radians (R_V1)
%
%=========================================================================================
% Calculation of "Centrifugal Force" at the root:
%-----------------------------------------------------------------------------------------
T_n(1)   = m_n(1) * R_n(1) * R_V1^2;              % ....... (T_n) and radians (R_V1)
%
% T_n(1) is is the "Centrifugal Force at the TIP"
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * R_n(i+1) * R_V1^2;
end;                                             % end of loop for i--> { T_n(i) }
%
disp( ' Centrifugal Force : ' )
disp(T_n(n))
% Displays "Centrifugal Force at r = 12.63"
%
%=========================================================================================
k        = 1;
i        = 1;
```

```
%
for omega_1 = 0 : modeshp^2*R_V/500 : 2.1*modeshp^2*R_V,
                                              % RPM (omega_1)
    omega(k) = omega_1*2*pi/60;               % radians (omega(k))
    F = eye(4);
    %------------------------------------------------------------------------------
    for j = 1:n-1,
        % determine length of the segments;
        l_sn = R_n(j) - R_n(j+1);
        %
        % detemine the stiffness
        EI = E_n * I_yy(j);
        %
        G_n = [ 1,     0,        0,          -m_n(j+1)*(omega(k)^2 + R_V1^2);
                0,     1,        0,          -T_n(j)             ;
                0,     0,        1,           0                  ;
                0,     0,        l_sn,        1                 ];

        A_n = [ 1,                 0,                 0,                        0      ;
                l_sn,              1,                 0,                       -T_n(j) ;
               -(l_sn^2)/(2*EI),  -l_sn/EI,           1+(T_n(j)*l_sn^2)/(2*EI), 0      ;
               -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),   (T_n(j)*l_sn^3)/(3*EI),   1     ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                                        % end of loop for j --> { A_n, G_n, F_n
)
    %
    %------------------------------------------------------------------------------
     B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);


    %
    %------------------------------------------------------------------------------
 %determination of the points and the natural frequencies where det crosses the "0 line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
   else
```

```
       if k >1,                                 % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                 % end of if (1.b)
       end;                                     % end of if (1.a)
   end;                                         % end of elseif (1)
   %
   k = k+1;
end;                                            % end of loop for omega_1
%
%================================================================================
%
% Sorting and Displaying the "Natural Frequencies"
%
% Sort and convert the unit of natural frequency to 'cpm'
for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;                                            % end of loop for (m)
%

%=================================================================================
%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);  %       % radians (omega_0) ,RPM (omega_n)
    %
    omega_new = 100;                          % omega_new i  in RPM
    omega_old = -100;                         % omega_old is in RPM
    flg_slope = 1;
    %
    %-------------------------------------------------------------------------------
    % Determination of odd or even mode shapes
    %
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
```

```
display('This is an incorrect mode shape input!')
end                                    % end of 'if loop R'
%
%---------------------------------------------------------------------------------
% Initial values
flag = 1;
flag_1 = 1;
det_bcold = 0;
det_bcnew = 0;
%
%--------------------------------------------------------------------------------
%
while flag == 1,
    F = eye(4);
    if omega_0 < 0,
       omega_0 = 0;                   % if natural frequency is negative, make it zero
    end                                  % end of if loop omega
    %
    for j = 1:n-1,
        %
        l_sn = R_n(j) - R_n(j+1);
        %
        EI = E_n * I_yy(j);
        %
        G_n=[1,     0,        0,        -m_n(j+1)*(omega_0^2 + R_V1^2)     ;
             0,     1,        0,        -T_n(j)                 ;
             0,     0,        1,         0                      ;
             0,     0,        l_sn,      1                      ];
        %
        A_n=[1,              0,                  0,                        0       ;
             l_sn,          1,                 0,                        -T_n(j)  ;
            -(l_sn^2)/(2*EI), -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),       0       ;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI),  (T_n(j)*l_sn^3)/(3*EI),  1       ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                          % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %
     B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
det_bc = det(B_c);
```

```
%--------------------------------------------------------------------------------
%
if (abs(det_bc)<10000 | abs(omega_old-omega_new)<1e-10),
                                    % if loop 1
    omega_n(count) = omega_0*30/pi; % radians (omega_0), RPM (omega_n)
    %
    if abs(omega_old-omega_new)<1e-10,                          % if loop 1.a
        disp('Warning!!  This value of omega may be in error.')
    end                                                  % end of 'if loop 1.a'
    %
    % change the initial values
    flag = 0;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %
else                                                       % else for if loop 1
    %
    if flag_1 == 1,                                              % if loop 1.b
        if det_bc > 0,                                                % if loop
1.b.i
            omega_new = omega_0 - (200*pi/30)*slope;    % radians(omega_new)
        else                                            % else for if loop 1.b.i
            omega_new = omega_0 + (200*pi/30)*slope;
        end;                                          % end of 'loop if 1.b.i'
        %
        % Rearrange the values
        %
        omega_old = omega_0;
        omega_0 = omega_new;
        det_bcold = det_bc;
        flag_1 = 2;
        %
    elseif flag_1 == 2,                               % elseif for if loop 1.b
        if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                                  % if loop 1.b.ii
            if det_bc > 0,                                 % if loop 1.b.ii.a
```

```
                        omega_new = omega_0 - (200*pi/30)*slope;
                    else                                    % else for if loop 1.b.ii.a
                        omega_new = omega_0 + (200*pi/30)*slope;
                    end;                                     % end of 'if loop 1.b.ii.a
                    %
                    % Rearrange the values
                    omega_old = omega_0;
                    omega_0 = omega_new;
                    det_bcold = det_bc;


                else                                         % else for if loop 1.b.ii
                    omega_0 = (omega_new + omega_old)/2;
                    det_bcnew = det_bc;
                    flag_1 = 0;
                end                                           % end of 'if loop 1.b.ii'
            else                                                 % else for if loop 1-b
                if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),% if loop
1.b.iii
                    omega_new = omega_0;
                    det_bcnew = det_bc;
                elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),%
elseif for if loop 1.b.ii
                    omega_old = omega_0;
                    det_bcold = det_bc;
                end;                                         % end of 'if loop 1.b.ii'
                %
                omega_0 = (omega_new + omega_old)/2;
                %
            end;                                                % end of 'if loop 1.b'
            %
        end;                                                      % end of 'if loop 1'
        %
    end;                                                         % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
 % According to the equation we can also use as : theta_n(count)= -B_c(2,2)/B_c(2,1)
 %
end                                                          % end of loop for 'count'
%
%==========================================================================================
%
```

```
for m= 1 : count                                                    % for loop 'm'
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                                  % for loop 'j'
        %
        l_sn    = R_n(j) - R_n(j+1);
        %
        EI      = E_n * I_yy(j);
        %
        G_n     =[ 1,      0,         0,       -m_n(j+1)*((omega_n(m)*pi/30)^2  + R_V1^2);
                   0,      1,         0,       -T_n(j)                         ;
                   0,      0,         1,        0                              ;
                   0,      0,       l_sn,             1                        ];
        %
        A_n     =[ 1,           0,              0,                         0       ;
                  l_sn,       1,                0,                        -T_n(j)  ;
                -(l_sn^2)/(2*EI), -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),  0       ;
               -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),    1       ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
        %
        X_n(:,j+1) = F * X_tip;
        %
    end;                             % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % Get the total deflection at each radial station
    %
    deflection(m,:)=X_n(4,:);
    plot ( R_n, 2*X_n(4,:), '-')

    %plot ( R_n, 2*X_n(4,:), '-');
    grid on;
    title ( ' Chordwise Mode Shapes at Operational Rotational Velocity ' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
hold off
```

```
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency/Rotational Speed is: ')
disp(omega_n/R_V)
%
% end of program "HINGELESS-EDGEWISE"
```

E. HINGED NONROTATING UNIFORM BLADE TEST

```
clear
clc
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Test of Hinged Nonrotating Uniform Beam                            %
%                     Corresponding Data Supported-Free Beam                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Uniform blade
R           = 372 ;
Ixx         = 238.23;
weight      = 316.6;
segment     = 372/50;
segment_I   = 238.23/50;
segment_W   = 316.6/50;
z = 1;
r_n         = R;
r_n(51)     = 0.00001;
I_xx(51)    = 0.0001;
Weight_n(51)= 0.0001;
I_xx(1)     = segment_I;
Weight_n(1) = segment_W;
%
for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
    I_xx(seg)       = segment_I;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;
%
E_n = 3e7;
%
```

```
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
omega = 0;
det_Bc = 0;
modeshp =input('How many mode shapes would you like?  ');
%*************************************************************************************
R_V = 0;                                 % Nonrotating Blade rotational speed should be 0
R_V1= R_V * pi/30;                       % R_V1 is in radians
T_n(1) =0;                               % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;

end;                                     % end of loop for i--> { T_n(i) }
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%**************************************************************************************
k = 1;
i = 1;
for omega1 = 0 :modeshp^2:2.1*modeshp^2*200,        % omega1 is in RPM
    omega(k) = omega1*2*pi/60;                      % omega(k) is in radians
    F = eye(4);
    %-----------------------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);


        EI = E_n * I_xx(j);



        G_n=[1,     0,        0,          -m_n(j+1)*omega(k)^2;
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];

        A_n=[1,                0,                  0,                 0;
             l_sn,             1,                  0,            -T_n(j);
```

```
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];


        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                          % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %-------------------------------------------------------------------------------


    B_c = [ F(2,3),F(2,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);


    %determination of the points and the natural frequencies where det crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
    else
       if k >1,                                % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                % end of if (1.b)
       end;                                    % end of if (1.a)
   end;                                        % end of elseif (1)
   %
   k = k+1;
end;                                           % end of loop for omega_1
%


for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                        % omega_new is  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    % determination of odd or even mode shapes
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end % end of 'if loop R'
    %----------------------------------------------
    % initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %-----------------------------------------------
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
            omega_0 = 0;% if real part of natural frequency is negative value make it
%zero
        end                       % end of if loop omega
        %------------------------------------------
        for j = 1:n-1,
            %
            l_sn = r_n(j) - r_n(j+1);
            %
            EI = E_n * I_xx(j);
            %
            G_n=[1,     0,        0,        -m_n(j+1)*omega_0^2   ;           % omega_0
is in radians
                 0,     1,        0,        -T_n(j)          ;
                 0,     0,        1,         0                 ;
                 0,     0,        l_sn,      1                 ];
            %
            A_n=[1,          0,              0,                        0       ;
              l_sn,          1,              0,                          -T_n(j);
          -(l_sn^2)/(2*EI), -l_sn/EI,          1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
```

```
         -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI),  (T_n(j)*l_sn^3)/(3*EI),       1       ];
          %
          F_n = inv(G_n)*A_n;
          F = F_n * F;
      end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
      %
      B_c = [ F(2,3), F(2,4) ; F(4,3), F(4,4) ];
      det_bc = det(B_c);
      %----------------------------------------------------------
      if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),          % if loop 1
          omega_n(count) = omega_0*30/pi;             % omega_0 is in radians, omega_n
is in RPM
          %
          %--------------------------------
          if abs(omega_old-omega_new)<1e-10,                           % if loop 1-a
              disp('Warning!!  This value of omega may be in error.')
          end                                                      % end of 'if loop 1-a'
          %-------------------------------
          % change the initial values
          flag = 0;
          flag_1 = 1;
          det_bcold = 0;
          det_bcnew = 0;
      else                                                           % else for if loop 1
          %-------------------------------
          if flag_1 == 1,                                             % if loop 1-b
              if det_bc > 0,                                         % if loop 1-b-i
                  omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
              else                                            % else for if loop 1-b-i
                  omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
              end;                                             % end of 'loop if 1-b-i'
              % rearrange the values
              omega_old = omega_0;
              omega_0 = omega_new;
              det_bcold = det_bc;
              flag_1 = 2;
          elseif flag_1 == 2,                             % elseif for if loop 1-b
              if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,    % if loop
1-b-ii
                  if det_bc > 0,                                   % if loop 1-b-ii-a
                      omega_new = omega_0 - (200*pi/30)*slope;
```

```
                                % omega_new is in radians
                else                                   % else for if loop 1-b-ii-a
                    omega_new = omega_0 + (200*pi/30)*slope;
                                       % omega_new is in radians
                end;                                   % end of 'if loop 1-b-ii-a
            % rearrange the values
            omega_old = omega_0;
            omega_0 = omega_new;
            det_bcold = det_bc;
            else                                           % else for if loop 1-b-ii
                omega_0  =  (omega_new  +  omega_old)/2;  %  omega_old,new  and  0  are  in
radians
                det_bcnew = det_bc;
                flag_1 = 0;
            end                                            % end of 'if loop 1-b-ii'
        else                                                   % else for if loop 1-b
            if  ((det_bc>0  &  det_bcnew  >  0)  |  (det_bc<0  &  det_bcnew  <  0)),    %  if
loop 1-b-iii
                omega_new = omega_0;
                det_bcnew = det_bc;
            elseif  ((det_bc  <  0  &  det_bcnew  >  0)  |  (det_bc  >  0  &  det_bcnew  <  0)),  %
elseif for if loop 1-b-ii
                omega_old = omega_0;
                det_bcold = det_bc;
            end;                                           % end of 'if loop 1-b-ii'
            omega_0 = (omega_new + omega_old)/2;
        end;                                                % end of 'if loop 1-b'
        %----------------------------------------
    end;                                                        % end of 'if loop 1'
    %------------------------------------------------------------------------------
end;                                                          % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% *****************************************************************************************
%
for m= 2 : count                                                        % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                                          % for loop j
```

```
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_xx(j);
        %
        G_n=[1,     0,        0,   -m_n(j+1)*(omega_n(m)*pi/30)^2;
            0,     1,        0,         -T_n(j);
            0,     0,        1,              0;
            0,     0,       l_sn,             1];

        A_n=[1,                0,                  0,                 0;
             l_sn,             1,                  0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
        X_n(:,j+1) = F * X_tip;
    end;                             % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    %
    plot ( r_n/r_n(1), 2*deflection(m,:), '-');
    grid on;
    axis([0 1 -2 2])
    title ( ' Hinged Nonrotating Uniform Blade Comparison (50 stations) ' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
hold on


% The University of Texas Publication uniform beam

% Clamped-Free Beams
%==========================================================================
Deflection_tx_mod3 = [  2.00000, 1.6861, 1.37287, 1.06189, 0.75558,...
                    0.45702, 0.16974, -0.10243, -0.35563, -0.58594,...
                    -0.78975,-0.96375, -1.10515, -1.21175, -1.28189,...
                    -1.31485, -1.31055, -1.26974, -1.19398, -1.08559,...
                    -0.94753, -0.78359, -0.59802, -0.39555, -0.18130,...
                    0.03937, 0.26103, 0.47822, 0.68568, 0.87841, ...
                    1.05185, 1.20196, 1.32534, 1.41931, 1.48203, ...
                    1.51248, 1.51046, 1.47707, 1.41376, 1.32324, ...
                    1.20901, 1.07535, 0.92728, 0.77049, 0.61120, ...
                    0.45614, 0.31238, 0.18727, 0.08829, 0.02339, ...
                    0];

Deflection_tx_mod2 = [2.0000, 1.80877, 1.61764, 1.42680, 1.23660, ...
                        1.04750, 0.86004, 0.67484, 0.49261, 0.31409,...
                        0.14007, -0.02865, -0.19123, -0.34687, -0.49475, ...
                        -0.63410, -0.76419, -0.88431, -0.99384, -1.09222, ...
                        -1.17895, -1.25365, -1.31600, -1.36578, -1.40289, ...
                        -1.42733, -1.43920, -1.43871, -1.42619, -1.40209, ...
                        -1.36694, -1.32141, -1.26626, -1.20236, -1.13068, ...
                        -1.05227, -0.96827, -0.87992, -0.78852, -0.69544, ...
                        -0.60211, -0.51002, -0.42070, -0.33573, -0.25670, ...
                        -0.18526, -0.12305, -0.07174, -0.03301, -0.00853,...
                        0];

Deflection_tx_mod1 = [2.0000, 1.94494, 1.88988, 1.83483, 1.77980,...
                        1.72480, 1.66985, 1.61496, 1.56016, 1.50549,...
                        1.45096, 1.39660, 1.34247, 1.28859, 1.23500,...
                        1.18175, 1.12889, 1.07646, 1.02451, 0.97309, ...
                        0.92227, 0.87209, 0.82262, 0.77392, 0.72603,...
                        0.67905, 0.63301, 0.58800, 0.54408, 0.50131,...
                        0.45977, 0.41952, 0.38065, 0.34322, 0.30730,...
                        0.27297, 0.24030, 0.20936, 0.18024, 0.15301, ...
                        0.12774, 0.10452, 0.08340, 0.06449, 0.04784,...
                        0.03355, 0.02168, 0.01231, 0.00552,0.00139,...
                        0];

%===============================================================================


% Clamped-Supported Beam
%===============================================================================
Deflection_tex_mod1 = [2.00000, 1.84282, 1.68568, 1.52869, 1.37202,...
```

```
                         1.21590, 1.06060, 0.90647, 0.75386, 0.60318,...
                         0.45486, 0.30935, 0.16712, 0.02866,-0.10554,...
                        -0.23500,-0.35923,-0.47775,-0.59009,-0.69582,...
                        -0.79450,-0.88574,-0.96918,-1.04447,-1.11133,...
                        -1.16950,-1.21875,-1.25894,-1.28992,-1.31162,...
                        -1.32402,-1.32714,-1.32106,-1.30588,-1.28180,...
                        -1.24904,-1.20786,-1.15858,-1.10157,-1.03725,...
                        -0.96606,-0.88849,-0.80507,-0.71636,-0.62295,...
                        -0.52547,-0.42455,-0.32086,-0.21507,-0.10789,...
                        0];

Deflection_tex_mod2 =  [2.00000, 1.71729, 1.43502, 1.15424, 0.87658,...
                         0.60415, 0.33937, 0.08494,-0.15633,-0.38158,...
                        -0.58802,-0.77300,-0.93412,-1.06927,-1.17673,...
                        -1.25518,-1.30380,-1.32224,-1.31068,-1.26983,...
                        -1.20092,-1.10569,-0.98634,-0.84553,-0.68631,...
                        -0.51204,-0.32640,-0.13323, 0.06348, 0.25968,...
                         0.45136, 0.63460, 0.80569, 0.96112, 1.09776,...
                         1.21281, 1.30395, 1.36930, 1.40755, 1.41789,...
                         1.40010, 1.35450, 1.28198, 1.18399, 1.06244,...
                         0.91976, 0.75879, 0.58271, 0.39504, 0.19951,...
                         0];

Deflection_tex_mod3 =  [2.00000, 1.59173, 1.18532, 0.78508, 0.39742,...
                         0.03009,-0.30845,-0.60968,-0.86560,-1.06927,...
                        -1.21523,-1.29988,-1.32158,-1.28137,-1.18195,...
                        -1.02863,-0.82967,-0.59110,-0.32637,-0.04596,...
                         0.23807, 0.51362, 0.76897, 0.99330, 1.17711,...
                         1.31263, 1.39411, 1.41807, 1.38344, 1.29160,...
                         1.14631, 0.95356, 0.72134, 0.45927, 0.17821,...
                        -0.11017,-0.39391,-0.66123,-0.90103,-1.10335,...
                        -1.25980,-1.36386,-1.41124,-1.39996,-1.33049,...
                        -1.20573,-1.03085,-0.81313,-0.56162,-0.28677,...
                        0];

  x_over_l(51) = 0;
  w = 51;
for q = 50:-1:1,
    x_over_l(q)= x_over_l(w) + 0.02;
    w = w-1;
```

```
end;
plot (x_over_l, Deflection_tex_mod3, 'r--')
hold on
plot (x_over_l, Deflection_tex_mod2, 'g--')
hold on
plot ( x_over_l, Deflection_tex_mod1, 'm--')

hold off

disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
%
% Comparison of the Natural Frequencies:
%
% HARTOG( Mechanical Vibrations - page:432):
% "Hinged-free" beam or wing of autogyro may be considered as half a "free-free" beam
%for even a-numbers
a_hartog = [0, 15.4, 50.0, 104, 178 ];
%
% Dana Young:
% Second derivative of Clamped-Supported is Free-Supported(hinged),
betal_young= [0, 15.418206, 49.964862, 104,247697, 178.269730 ];
% Our Program:
if modeshp > 5
    com = 5;
else
    com = modeshp;
end
%
for i = 1:(com-1);
    compare (i) = omega_n(i)/omega_n(i+1);
    a_hartog_compare(i) = a_hartog(i) / a_hartog(i+1);
    betal_young_compare(i) = betal_young(i) / betal_young(i+1);
end
display ( ' According to Hartog the coefficients are: ')
disp( a_hartog_compare)
display ( ' According to Young and Felgar the coefficients are: ')
disp( betal_young_compare)
display ( ' According to Our Program the coefficients are: ')
```

```
disp( compare)
%
for j = 1:(com-1)
    if betal_young_compare(j) == 0;
        accuracy(j) = compare(j) * 100;
    else
    accuracy (j) = (compare(j) / betal_young_compare(j) - 1) *100;
    end
end
disp ( 'The Error of the Natural Frequencies(%) ' )
disp ( accuracy)
%
% end of program "HINGED-TEST"
```

F. HINGELESS NONROTATING UNIFORM BLADE TEST

```
clear
clc
format long
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Hingeless Nonrotating Uniform Blade Comparison                          %
%               Corresponding Data is Clamped-Free Beam                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Uniform blade
R           = 372 ;
Ixx         = 58.23;
weight      = 316.6;
segment     = 372/50;
segment_I   = 58.23/50;
segment_W   = 316.6/50;
z = 1;
r_n         = R;
r_n(51)     = 0.00001;
I_xx(51)    = 0.0001;
Weight_n(51)= 0.0001;
I_xx(1)     = segment_I;
Weight_n(1) = segment_W;
%
for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
    I_xx(seg)       = segment_I;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;
%
E_n = 3e7;
%
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
omega = 0;
det_Bc = 0;
modeshp =input('How many mode shapes would you like?  ');
%*************************************************************************************
R_V = 0;                                        % Operational Velocity should be zero
R_V1= R_V * pi/30;                              % R_V1 is in radians
T_n(1) = m_n(1) * r_n(1) * R_V1^2;              % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;

end; % end of loop for i--> { T_n(i) }
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%*************************************************************************************
k = 1;
i = 1;
for omega1 = 0 : modeshp^2 : 2.1*modeshp^2*200,      % omega1 is in RPM
    omega(k) = omega1*2*pi/60;          % omega(k) is in radians
    F = eye(4);
    %--------------------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);



        EI = E_n * I_xx(j);
        G_n=[1,     0,        0,          -m_n(j+1)*omega(k)^2;
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];

        A_n=[1,                 0,                  0,                 0;
             l_sn,              1,                  0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                             % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %-----------------------------------------------------------------------------------

    B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);

       if k >1,
       s = k-1;
       if det_bc(k) * det_bc(:,s) < 0
       omega_natural(i) = (omega(k) + omega(:,s))/2;
       i = i +1;
   end;
   end;

k = k+1;
end; % end of loop for omega1
%

for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                        % omega_new is  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
```

```
%
% determination of odd or even mode shapes
R = rem(count,2);
if R ==1
slope = 1;
elseif R == 0;
slope = -1;
else
display('This is an incorrect mode shape input!')
end % end of 'if loop R'
%----------------------------------------------
% initial values
flag = 1;
flag_1 = 1;
det_bcold = 0;
det_bcnew = 0;
%-----------------------------------------------
%
while flag == 1,
    F = eye(4);
    if omega_0 < 0,
        omega_0 = 0;
         % if real part of natural frequency is negative value make it zero
    end                 % end of if loop omega
    %-----------------------------------------
    for j = 1:n-1,
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_xx(j);
        %
        G_n=[1,     0,        0,       -m_n(j+1)*omega_0^2  ;% omega_0 is in radians
             0,     1,        0,        -T_n(j)           ;
             0,     0,        1,         0                  ;
             0,     0,        l_sn,      1                  ];
        %
        A_n=[1,                0,                 0,                         0       ;
           l_sn,            1,                 0,                            -T_n(j);
        -(l_sn^2)/(2*EI),   -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
       -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),    (T_n(j)*l_sn^3)/(3*EI),     1     ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    B_c = [ F(3,3), F(3,4) ; F(4,3), F(4,4) ];
    det_bc = det(B_c);
    %----------------------------------------------------------
    if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),         % if loop 1
        omega_n(count) = omega_0*30/pi;     omega_0 is in radians, omega_n is in RPM
        %
        %--------------------------------
        if abs(omega_old-omega_new)<1e-10,                        % if loop 1-a
            disp('Warning!!  This value of omega may be in error.')
        end                                              % end of 'if loop 1-a'
        %-------------------------------
        % change the initial values
        flag = 0;
        flag_1 = 1;
        det_bcold = 0;
        det_bcnew = 0;
    else                                                      % else for if loop 1
        %-------------------------------
        if flag_1 == 1,                                             % if loop 1-b
            if det_bc > 0,                                          % if loop 1-b-i
                omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
            else                                         % else for if loop 1-b-i
                omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
            end;                                         % end of 'loop if 1-b-i'
            % rearrange the values
            omega_old = omega_0;
            omega_0 = omega_new;
            det_bcold = det_bc;
            flag_1 = 2;
        elseif flag_1 == 2,                               % elseif for if loop 1-b
            if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                           % if loop 1-b-ii
                if det_bc > 0,                                  % if loop 1-b-ii-a
                    omega_new = omega_0 - (200*pi/30)*slope;
                                           % omega_new is in radians
```

```
                  else                                    % else for if loop 1-b-ii-a
                      omega_new  =  omega_0  +  (200*pi/30)*slope;  %  omega_new  is  in
radians
                  end;                                     % end of 'if loop 1-b-ii-a
              % rearrange the values
              omega_old = omega_0;
              omega_0 = omega_new;
              det_bcold = det_bc;
              else                                          % else for if loop 1-b-ii
                  omega_0  =  (omega_new  +  omega_old)/2;  %  omega_old,new  and  0  are  in
radians
                  det_bcnew = det_bc;
                  flag_1 = 0;
              end                                           % end of 'if loop 1-b-ii'
          else                                               % else for if loop 1-b
              if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
                                                    % if loop 1-b-iii
                  omega_new = omega_0;
                  det_bcnew = det_bc;
              elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),
                                             % elseif for if loop 1-b-ii
                  omega_old = omega_0;
                  det_bcold = det_bc;
              end;                                         % end of 'if loop 1-b-ii'
              omega_0 = (omega_new + omega_old)/2;
          end;                                                 % end of 'if loop 1-b'
          %---------------------------------------
      end;                                                     % end of 'if loop 1'
      %------------------------------------------------------------------------------
  end;                                                       % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% *************************************************************************************
%
for m= 1 : count                                                   % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                                  % for loop j
        %
      l_sn = r_n(j) - r_n(j+1);
      %
      EI = E_n * I_xx(j);
      %
      G_n=[1,     0,        0,   -m_n(j+1)*(omega_n(m)*pi/30)^2;
          0,     1,        0,         -T_n(j);
          0,     0,        1,              0;
          0,     0,       l_sn,             1];

      A_n=[1,                0,                 0,                 0;
           l_sn,             1,                 0,            -T_n(j);
          -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
          -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

      F_n = inv(G_n)*A_n;
      F = F_n * F;
      X_n(:,j+1) = F * X_tip;
    end;                              % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    plot ( r_n/r_n(1), 2*deflection(m,:), '-');
    grid on;
    axis([0 1 -2 2])
    title ( ' Hingeless Nonrotating Uniform Blade Comparison (50 stations) ' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
hold on


% The University of Texas Publication uniform beam

% Clamped-Free Beams
%============================================================================
Deflection_tx_mod3 = [  2.00000, 1.6861, 1.37287, 1.06189, 0.75558,...
                  0.45702, 0.16974, -0.10243, -0.35563, -0.58594,...
                  -0.78975,-0.96375, -1.10515, -1.21175, -1.28189,...
                  -1.31485, -1.31055, -1.26974, -1.19398, -1.08559,...
```

```
                -0.94753, -0.78359, -0.59802, -0.39555, -0.18130,...
                0.03937, 0.26103, 0.47822, 0.68568, 0.87841, ...
                1.05185, 1.20196, 1.32534, 1.41931, 1.48203, ...
                1.51248, 1.51046, 1.47707, 1.41376, 1.32324, ...
                1.20901, 1.07535, 0.92728, 0.77049, 0.61120, ...
                0.45614, 0.31238, 0.18727, 0.08829, 0.02339, ...
                0];

Deflection_tx_mod2 = [2.0000, 1.80877, 1.61764, 1.42680, 1.23660, ...
                    1.04750, 0.86004, 0.67484, 0.49261, 0.31409,...
                    0.14007, -0.02865, -0.19123, -0.34687, -0.49475, ...
                    -0.63410, -0.76419, -0.88431, -0.99384, -1.09222, ...
                    -1.17895, -1.25365, -1.31600, -1.36578, -1.40289, ...
                    -1.42733, -1.43920, -1.43871, -1.42619, -1.40209, ...
                    -1.36694, -1.32141, -1.26626, -1.20236, -1.13068, ...
                    -1.05227, -0.96827, -0.87992, -0.78852, -0.69544, ...
                    -0.60211, -0.51002, -0.42070, -0.33573, -0.25670, ...
                    -0.18526, -0.12305, -0.07174, -0.03301, -0.00853,...
                    0];

Deflection_tx_mod1 = [2.0000, 1.94494, 1.88988, 1.83483, 1.77980,...
                    1.72480, 1.66985, 1.61496, 1.56016, 1.50549,...
                    1.45096, 1.39660, 1.34247, 1.28859, 1.23500,...
                    1.18175, 1.12889, 1.07646, 1.02451, 0.97309, ...
                    0.92227, 0.87209, 0.82262, 0.77392, 0.72603,...
                    0.67905, 0.63301, 0.58800, 0.54408, 0.50131,...
                    0.45977, 0.41952, 0.38065, 0.34322, 0.30730,...
                    0.27297, 0.24030, 0.20936, 0.18024, 0.15301, ...
                    0.12774, 0.10452, 0.08340, 0.06449, 0.04784,...
                    0.03355, 0.02168, 0.01231, 0.00552,0.00139,...
                    0];

%===============================================================================


% Clamped-Supported Beam
%===============================================================================
Deflection_tex_mod1 = [2.00000, 1.84282, 1.68568, 1.52869, 1.37202,...
                    1.21590, 1.06060, 0.90647, 0.75386, 0.60318,...
                    0.45486, 0.30935, 0.16712, 0.02866,-0.10554,...
```

```
                       -0.23500,-0.35923,-0.47775,-0.59009,-0.69582,...
                       -0.79450,-0.88574,-0.96918,-1.04447,-1.11133,...
                       -1.16950,-1.21875,-1.25894,-1.28992,-1.31162,...
                       -1.32402,-1.32714,-1.32106,-1.30588,-1.28180,...
                       -1.24904,-1.20786,-1.15858,-1.10157,-1.03725,...
                       -0.96606,-0.88849,-0.80507,-0.71636,-0.62295,...
                       -0.52547,-0.42455,-0.32086,-0.21507,-0.10789,...
                       0];

Deflection_tex_mod2 =  [2.00000, 1.71729, 1.43502, 1.15424, 0.87658,...
                        0.60415, 0.33937, 0.08494,-0.15633,-0.38158,...
                       -0.58802,-0.77300,-0.93412,-1.06927,-1.17673,...
                       -1.25518,-1.30380,-1.32224,-1.31068,-1.26983,...
                       -1.20092,-1.10569,-0.98634,-0.84553,-0.68631,...
                       -0.51204,-0.32640,-0.13323, 0.06348, 0.25968,...
                        0.45136, 0.63460, 0.80569, 0.96112, 1.09776,...
                        1.21281, 1.30395, 1.36930, 1.40755, 1.41789,...
                        1.40010, 1.35450, 1.28198, 1.18399, 1.06244,...
                        0.91976, 0.75879, 0.58271, 0.39504, 0.19951,...
                        0];

Deflection_tex_mod3 =  [2.00000, 1.59173, 1.18532, 0.78508, 0.39742,...
                        0.03009,-0.30845,-0.60968,-0.86560,-1.06927,...
                       -1.21523,-1.29988,-1.32158,-1.28137,-1.18195,...
                       -1.02863,-0.82967,-0.59110,-0.32637,-0.04596,...
                        0.23807, 0.51362, 0.76897, 0.99330, 1.17711,...
                        1.31263, 1.39411, 1.41807, 1.38344, 1.29160,...
                        1.14631, 0.95356, 0.72134, 0.45927, 0.17821,...
                       -0.11017,-0.39391,-0.66123,-0.90103,-1.10335,...
                       -1.25980,-1.36386,-1.41124,-1.39996,-1.33049,...
                       -1.20573,-1.03085,-0.81313,-0.56162,-0.28677,...
                       0];

  x_over_l(51) = 0;
  w = 51;
for q = 50:-1:1,
    x_over_l(q)= x_over_l(w) + 0.02;
    w = w-1;
end;
plot (x_over_l, Deflection_tx_mod3, 'r--')
```

```
hold on
plot (x_over_l, Deflection_tx_mod2, 'g--')
hold on
plot ( x_over_l, Deflection_tx_mod1, 'm--')

hold off
 %
 %==================================================================================
%
% Comparison of the Natural Frequencies:
%
% HARTOG( Mechanical Vibrations - page:432):
% "Hinged-free" beam or wing of autogyro may be considered as half a "free-free" beam
%for even a-numbers
a_hartog = [3.52, 22.0, 61.7, 121.0, 200];
%
% Dana Young:
% Second derivative of Clamped-Supported is Free-Supported(hinged),
betal_young= [3.5150154, 22.034492, 61.697214, 120.901916, 199.859530];
% Our Program:
if modeshp > 5
    com = 5;
else
    com = modeshp;
end
%
for i = 1:(com-1);
    compare (i) = omega_n(i)/omega_n(i+1);
    a_hartog_compare(i) = a_hartog(i) / a_hartog(i+1);
    betal_young_compare(i) = betal_young(i) / betal_young(i+1);
end
display ( ' According to Hartog the coefficients are: ')
disp( a_hartog_compare)
display ( ' According to Dana Youn the coefficients are: ')
disp( betal_young_compare)
display ( ' According to Our Program the coefficients are: ')
disp( compare)
%
%
for j = 1:(com-1)
```

```
    if betal_young_compare(j) == 0;
        accuracy(j) = compare(j) * 100;
    else
    accuracy (j) = (compare(j) / betal_young_compare(j) - 1) *100;
    end
end
disp ( 'The Accuracy of the Natural Frequencies(%) ' )
disp ( accuracy)
%
% end of program "HINGELESS-TEST"
```

G. GUI PROGRAMS

1. Call function file of the Gui page

```
function varargout = finalthesis(varargin)
% FINALTHESIS Application M-file for finalthesis.fig
%    FIG = FINALTHESIS launch finalthesis GUI.
%    FINALTHESIS('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 05-Sep-2002 13:06:07

if nargin == 0  % LAUNCH GUI

  fig = openfig(mfilename,'reuse');

  % Use system color scheme for figure:
  set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

  % Generate a structure of handles to pass to callbacks, and store it.
  handles = guihandles(fig);
  guidata(fig, handles);



%========================================================================
% Call the popup menu callback to initialize the handles.data
% Field with the current value of the popup
%
% (1) Popup Menu "en"
en_popupmenu_Callback(handles.en_popupmenu,[],handles)
```

```
%
% (2) Popup Menu "mode shape"
mode shape_popupmenu_Callback(handles.mode shape_popupmenu,[],handles)
%
% (3) Popup Menu "hinge"
hinge_popupmenu_Callback(handles.hinge_popupmenu,[],handles)
%
% (4) Popup Menu "axis"
axis_popupmenu_Callback(handles.axis_popupmenu,[],handles)
%
% (5) Popup Menu "form"
form_popupmenu_Callback(handles.form_popupmenu,[],handles)
%
% ==========================================================================




  if nargout > 0
        varargout{1} = fig;
  end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

  try
        if (nargout)
              [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
              feval(varargin{:}); % FEVAL switchyard
        end
  catch
        disp(lasterr);
  end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.



% --------------------------------------------------------------------
function varargout = en_popupmenu_Callback(h, eventdata, handles, varargin)
%
val = get(h,'Value');
%
switch val
```

```
case 1 % user selected 'aliminum'
    E_n = 1e7;
    handles.en = E_n;

case 2 % user selected 'composite'
    E_n = 2e7;
    handles.en = E_n;

case 3 % user selected 'steel'
    E_n = 3e7;
    handles.en = E_n;

case 4 % user selected 'titanium'
    E_n = 2.5e7;
    handles.en = E_n;

end
%
guidata(h,handles) % Save the handles structure after adding  data
%



% --------------------------------------------------------------------
function varargout = mode shape_popupmenu_Callback(h, eventdata, handles, varargin)
%
val = get(h,'Value');

switch val

case 1 % user selected '1 mode shape'
    modeshp = 1;
    handles.mode shape = modeshp;

case 2 % user selected '2 mode shape'
    modeshp = 2;
    handles.mode shape = modeshp;

case 3 % user selected '3 mode shape'
    modeshp = 3;
    handles.mode shape = modeshp;

case 4 % user selected '4 mode shape'
    modeshp = 4;
    handles.mode shape = modeshp;

case 5 % user selected '5 mode shape'
    modeshp = 5;
    handles.mode shape = modeshp;

end
%
guidata(h,handles) % Save the handles structure after adding data
%
% --------------------------------------------------------------------
function varargout = hinge_popupmenu_Callback(h, eventdata, handles, varargin)
%
val = get(h,'Value');

%
switch val

case 1 % user selected 'hinged'
    hinge_option = 1;
    handles.hinge = hinge_option;

case 2 % user selected 'hingeless'
    hinge_option = 2;
    handles.hinge = hinge_option;

end
%
guidata(h,handles) % save the handles structure after adding data
%

% --------------------------------------------------------------------
function varargout = axis_popupmenu_Callback(h, eventdata, handles, varargin)
%
val = get(h,'Value');
%
```

```
switch val

case 1 % user selected 'flatwise'
    axis_option = 1;
    handles.axis = axis_option;

case 2 % user selected 'edgewise'
    axis_option = 2;
    handles.axis = axis_option;

case 3 % user selected 'coupled'
    axis_option = 3;
    handles.axis = axis_option;

end
%
guidata(h,handles) % save the handles structure after adding data
%
% --------------------------------------------------------------------
function varargout = form_popupmenu_Callback(h, eventdata, handles, varargin)
%
val = get(h,'Value');
%
switch val

case 1 % user selected 'uniform'
    form_option = 1;
    handles.form = form_option;

case 2 % user selected 'hingeless'
    form_option = 2;
    handles.form = form_option;

end
%
guidata(h,handles) % save the handles structure after adding data
%

% --------------------------------------------------------------------
```

```
function varargout = rpm_edit_Callback(h, eventdata, handles, varargin)
%
R_V = str2num(get(h,'String'));
handles.rpm = R_V;
%
guidata(h,handles); % save the handles structure after adding data
%
% --------------------------------------------------------------------
function varargout = e_edit_Callback(h, eventdata, handles, varargin)
%
e = str2num(get(h,'String'));
handles.e = e;
%
guidata(h,handles);
%

% --------------------------------------------------------------------
function varargout = radius_edit_Callback(h, eventdata, handles, varargin)
%
R = str2num(get(h,'String'));
handles.radius = R;
%
guidata(h,handles);
%
% --------------------------------------------------------------------
function varargout = weight_edit_Callback(h, eventdata, handles, varargin)
%
weight = str2num(get(h,'String'));
handles.weight = weight;
%
guidata(h,handles);
%
% --------------------------------------------------------------------
function varargout = inertia_edit_Callback(h, eventdata, handles, varargin)
%
I_xx = str2num(get(h,'String'));
handles.inertia = I_xx;
%
guidata(h,handles);
%
```

```
% --------------------------------------------------------------------
function varargout = cancel_pushbutton_Callback(h, eventdata, handles, varargin)
% --------------------------------------------------------------------
function varargout = ok_pushbutton_Callback(h, eventdata, handles, varargin)


%
E_n             = handles.en;
%
modeshp         = handles.mode shape;
%
hinge_option    = handles.hinge;
%
axis_option     = handles.axis;
%
form_option     = handles.form;
%
%
%
R_V             = handles.rpm;
%
e               = handles.e;
%
R               = handles.radius;
%
weight          = handles.weight;
%
I_xx            = handles.inertia;
%
%
%=========================================================================================
if      hinge_option == 2 & e ~= 0
            errordlg('Hingeless  Blade  Should  Have  0  Hinge-Offset!',  '  Offset  Error
Dialog Box','modal')
else
%
%=========================================================================================
if      hinge_option == 1 & e == 0
            errordlg('Hinged  Blade  Should  Have  Hinge-Offset  Value!','  Offset  Error
Dialog Box2','modal')
else
 %
```

```
%=========================================================================================
if        length(R) ~= length(weight) | length(R) ~= length(I_xx) | length(weight) ~=
length(I_xx)
            errordlg('The  Number  of  Stations  Should  Match!','Input  Error  Dialog
Box','modal')
else
%
%=========================================================================================
if      form_option == 1 & length(R) >1
            warndlg('Use More Than 1 Stations for Only Nonuniform Blades!','Blade Form
Input Warning Dialog Box','modal')
else
%
%=========================================================================================
if      R_V < 0 | e < 0 | R < 0 | weight < 0 | I_xx < 0
            errordlg('Invalid Input Value', 'Invalid Input Error Dialog Box','modal')
else
%
%=========================================================================================
reset(gca)
%
[cpm,ratio,tn] = operate( E_n, modeshp, hinge_option, axis_option, form_option, R_V, e,
R, weight, I_xx );
%
%
%
if length(cpm) == 1;

    value1 = cpm(1);
    str1 = sprintf('%.2f',value1);
    set (handles.cpm1_text,'String',str1);
%
    value2 = 0;
    str2 = sprintf('%.2f',value2);
    set (handles.cpm2_text,'String',str2);
%
    value3 = 0;
    str3 = sprintf('%.2f',value3);
    set (handles.cpm3_text,'String',str3);
%
    value4 = 0;
    str4 = sprintf('%.2f',value4);
    set (handles.cmp4_text,'String',str4);
%
    value5 = 0;
    str5 = sprintf('%.2f',value5);
    set (handles.cpm5_text,'String',str5);
end
%-----------------------------------------
if length(cpm) == 2;
    value1 = cpm(1);
    str1 = sprintf('%.2f',value1);
    set (handles.cpm1_text,'String',str1);
%
    value2 = cpm(2);
    str2 = sprintf('%.2f',value2);
    set (handles.cpm2_text,'String',str2);
%
    value3 = 0;
    str3 = sprintf('%.2f',value3);
    set (handles.cpm3_text,'String',str3);
%
    value4 = 0;
    str4 = sprintf('%.2f',value4);
    set (handles.cmp4_text,'String',str4);
%
    value5 = 0;
    str5 = sprintf('%.2f',value5);
    set (handles.cpm5_text,'String',str5);
end
%-----------------------------------------
if length(cpm) == 3
    value1 = cpm(1);
    str1 = sprintf('%.2f',value1);
    set (handles.cpm1_text,'String',str1);
%
    value2 = cpm(2);
    str2 = sprintf('%.2f',value2);
    set (handles.cpm2_text,'String',str2);
%
    value3 = cpm(3);
    str3 = sprintf('%.2f',value3);
    set (handles.cpm3_text,'String',str3);
%
    value4 = 0;
    str4 = sprintf('%.2f',value4);
    set (handles.cmp4_text,'String',str4);
%
    value5 = 0;
    str5 = sprintf('%.2f',value5);
    set (handles.cpm5_text,'String',str5);
end
%-------------------------------------------
if length(cpm) == 4
    value1 = cpm(1);
    str1 = sprintf('%.2f',value1);
    set (handles.cpm1_text,'String',str1);
%
    value2 = cpm(2);
    str2 = sprintf('%.2f',value2);
    set (handles.cpm2_text,'String',str2);
%
    value3 = cpm(3);
    str3 = sprintf('%.2f',value3);
    set (handles.cpm3_text,'String',str3);
%
    value4 = cpm(4);
    str4 = sprintf('%.2f',value4);
    set (handles.cmp4_text,'String',str4);
%
    value5 = 0;
    str5 = sprintf('%.2f',value5);
    set (handles.cpm5_text,'String',str5);
end
%-------------------------------------------
if length(cpm) == 5
    value1 = cpm(1);
    str1 = sprintf('%.2f',value1);
    set (handles.cpm1_text,'String',str1);
%
    value2 = cpm(2);
    str2 = sprintf('%.2f',value2);
    set (handles.cpm2_text,'String',str2);
%
    value3 = cpm(3);
    str3 = sprintf('%.2f',value3);
    set (handles.cpm3_text,'String',str3);
%
    value4 = cpm(4);
    str4 = sprintf('%.2f',value4);
    set (handles.cmp4_text,'String',str4);
%
    value5 = cpm(5);
    str5 = sprintf('%.2f',value5);
    set (handles.cpm5_text,'String',str5);
end
%
%================================================================
if length(ratio) == 1
%
val1 = ratio(1);
strg1 = sprintf('%.2f',val1);
set (handles.ratio1_text,'String',strg1);
%
val2 = 0;
strg2 = sprintf('%.2f',val2);
set (handles.ratio2_text,'String',strg2);
%
val3 = 0;
strg3 = sprintf('%.2f',val3);
set (handles.ratio3_text,'String',strg3);
%
val4 = 0;
strg4 = sprintf('%.2f',val4);
set (handles.ratio4_text,'String',strg4);
%
val5 = 0;
strg5 = sprintf('%.2f',val5);
set (handles.ratio5_text,'String',strg5);
end
%--------------------------------------------
```

```
if length(ratio) == 2
%
val1 = ratio(1);
strg1 = sprintf('%.2f',val1);
set (handles.ratio1_text,'String',strg1);
%
val2 = ratio(2);
strg2 = sprintf('%.2f',val2);
set (handles.ratio2_text,'String',strg2);
%
val3 = 0;
strg3 = sprintf('%.2f',val3);
set (handles.ratio3_text,'String',strg3);
%
val4 = 0;
strg4 = sprintf('%.2f',val4);
set (handles.ratio4_text,'String',strg4);
%
val5 = 0;
strg5 = sprintf('%.2f',val5);
set (handles.ratio5_text,'String',strg5);
end
%---------------------------------------------
if length(ratio) == 3
%
val1 = ratio(1);
strg1 = sprintf('%.2f',val1);
set (handles.ratio1_text,'String',strg1);
%
val2 = ratio(2);
strg2 = sprintf('%.2f',val2);
set (handles.ratio2_text,'String',strg2);
%
val3 = ratio(3);
strg3 = sprintf('%.2f',val3);
set (handles.ratio3_text,'String',strg3);
%
val4 = 0;
strg4 = sprintf('%.2f',val4);
set (handles.ratio4_text,'String',strg4);
```

```
%
val5 = 0;
strg5 = sprintf('%.2f',val5);
set (handles.ratio5_text,'String',strg5);
end
%
%---------------------------------------------
if length(ratio) == 4
%
val1 = ratio(1);
strg1 = sprintf('%.2f',val1);
set (handles.ratio1_text,'String',strg1);
%
val2 = ratio(2);
strg2 = sprintf('%.2f',val2);
set (handles.ratio2_text,'String',strg2);
%
val3 = ratio(3);
strg3 = sprintf('%.2f',val3);
set (handles.ratio3_text,'String',strg3);
%
val4 = ratio(4);
strg4 = sprintf('%.2f',val4);
set (handles.ratio4_text,'String',strg4);
%
val5 = 0;
strg5 = sprintf('%.2f',val5);
set (handles.ratio5_text,'String',strg5);
end
%
%------------------------------------------
if length(ratio) == 5%
%
val1 = ratio(1);
strg1 = sprintf('%.2f',val1);
set (handles.ratio1_text,'String',strg1);
%
val2 = ratio(2);
strg2 = sprintf('%.2f',val2);
set (handles.ratio2_text,'String',strg2);
```

```
%
val3 = ratio(3);
strg3 = sprintf('%.2f',val3);
set (handles.ratio3_text,'String',strg3);
%
val4 = ratio(4);
strg4 = sprintf('%.2f',val4);
set (handles.ratio4_text,'String',strg4);
%
val5 = ratio(5);
strg5 = sprintf('%.2f',val5);
set (handles.ratio5_text,'String',strg5);
end

%
%====================================================================
value_tn = tn;
strin = sprintf('%.2f',value_tn);
set(handles.tn_text,'String',strin);



%====================================================================
end     % ifelse for 'hingeless'
end     % ifelse for 'hinged'
end     % ifelse for 'inputs'
end     % ifelse for 'form inputs'
end     % ifelse for 'invalid inputs'
%
% end of program "FINALTHESIS"
```

2. Operation Function

```
          function  [cpm,ratio,tn,varargout]  =  operate(  E_n,  modeshp,  hinge_option,
axis_option, form_option, R_V, e, R, weight, I_xx)
       %
       %=====================================================================
=======
       %  This  function  is  the  main  operating  function  of  the  gui  program  of  the
'Mykelstad Blade Dynamics Analysis'
       %
       %=====================================================================
======
       %
       %---------------------------------------------------------------
       % Assigning the default values of the program:
       % (All the default values are given for the H-3 Helicopter values with no hinge)
       %
       %*
       %global cpm
       %global ratio
       %global tn
       %
       if isempty(R_V)
          R_V = 372;
       end
       %
       %**
       if isempty(e)
          e = 0;
       end
       %
       %***
       if isempty(R)
          R = 372;
       end
       %
       %****
       if isempty(weight)
          weight = 316.6;
       end
       %
       %*****
       if isempty(I_xx)
          I_xx = 2.5;
       end
       %
       %------------------------------------------------------------
       %
       %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
      switch form_option;                                 % switch(1)

      case 1                                              % 'uniform'


          switch axis_option;                                 % switch(2)

          case 1                                              % 'flatwise'

              switch hinge_option                                 % switch(3)
              case 1                                              % 'hinged'
                  [cpm,ratio,tn]  =  hinged_uniform_flatwise(E_n,  modeshp,  R_V,  e,  R,
weight,I_xx);
              case 2                                              % 'hingeless'
                  [cpm,ratio,tn] = hingeless_uniform_flatwise(E_n, modeshp, R_V, e, R,
weight,I_xx);
              end                                                 % end switch(3)
          %-----------------------



          case 2;                                            % 'edgewise'
              switch hinge_option                                 % switch(3)
              case 1                                              % 'hinged'
                  [cpm,ratio,tn]  =  hinged_uniform_edgewise(E_n,  modeshp,  R_V,  e,  R,
weight, I_xx);

              case 2                                              % 'hingeless'
                  [cpm,ratio,tn] = hingeless_uniform_edgewise(E_n, modeshp, R_V, e, R,
weight, I_xx);
              end                                                 % end switch(3)
          %-----------------------



          case 3;                                             % 'coupled'
              switch hinge_option                                       % switch(3)
              case 1                                              % 'hinged'
                  [cpm,ratio,tn]  =  hinged_uniform_coupled(E_n,  modeshp,  R_V,  e,  R,
weight, I_xx);
              case 2                                              % 'hingeless'
```

```
                   [cpm,ratio,tn] = hingeless_uniform_coupled(E_n, modeshp, R_V, e, R,
weight, I_xx);
               end                                                 % end switch(3)


           end                                                 % end switch(2)
           %-----------------------





       case 2                                              % ' nonuniform'


            switch axis_option;                                % switch(2)


               case 1                                          % 'flatwise'
                   switch hinge_option                                   % switch(3)
                   case 1                                          % 'hinged'
                       [cpm,ratio,tn] = hinged_nonuniform_flatwise(E_n, modeshp, R_V, e,
R, weight, I_xx);
                   case 2                                          % 'hingeless'
                       [cpm,ratio,tn] = hingeless_nonuniform_flatwise(E_n, modeshp, R_V,
e, R, weight, I_xx);
                   end                                             % end switch(3)
           %-----------------------



               case 2;                                         % 'edgewise'
                   switch hinge_option                                 % switch(3)
                   case 1                                              % 'hinged'
                       [cpm,ratio,tn] = hinged_nonuniform_edgewise(E_n, modeshp, R_V, e,
R, weight, I_xx);
                   case 2                                              % 'hingeless'
                       [cpm,ratio,tn] = hingeless_nonuniform_edgewise(E_n, modeshp, R_V,
e, R, weight, I_xx);
                   end                                                 % end switch(3)
           %------------------------



               case 3;                                         % 'coupled'
                   switch hinge_option                                 % switch(3)
                   case 1                                              % 'hinged'
```

```
                    [cpm,ratio,tn] = hinged_nonuniform_coupled(E_n, modeshp, R_V, e,
R, weight, I_xx);
                case 2                                              % 'hingeless'
                    [cpm,ratio,tn] = hingeless_nonuniform_coupled(E_n, modeshp, R_V,
e, R, weight, I_xx);
                end                                                 % end switch(3)
            %------------------------



            end                                                 % end switch(2)






        end                                                 % end switch(1)




            % end of the function "OPERATE"
```

3. Hinged Uniform Flatwise Function

```
function   [omega_n,omega_rate,cent_force,varargout]   =   hinged_uniform_flatwise(E_n,
modeshp, R_V, e, R, weight,I_xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           This function if for the uniform hinged blades                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
segment     = R/50;
segment_W   = weight/50;
z           = 1;
r_n(1)      = R;
r_n(51)     = e;
I_x(51)     = 0.0001;
Weight_n(51)= 0.0001;
I_x(1)      = I_xx;
Weight_n(1) = segment_W;

for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
        if r_n(seg) < e
        r_n(seg) = e;
        end
    I_x(seg)       = I_xx;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;
%
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
R_V1= R_V * pi/30;                             % R_V1 is in radians
T_n(1)  = 0;                                    % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;

end; % end of loop for i--> { T_n(i) }
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%*************************************************************************************
k = 1;
i = 1;
for omega1 = 0 :modeshp^2:2.1*modeshp^2*200,      % omega1 is in RPM
    omega(k) = omega1*2*pi/60;          % omega(k) is in radians
    F = eye(4);
    %------------------------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);



        EI = E_n * I_x(j);


        G_n=[1,     0,        0,          -m_n(j+1)*omega(k)^2;   omega(k) is in radians
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];


        A_n=[1,                 0,                  0,                  0;
             l_sn,              1,                  0,            -T_n(j);
```

```
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %----------------------------------------------------------------------------------

    B_c = [ F(2,3),F(2,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);



    %determination of the points and the natural frequencies where det crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
    else
       if k >1,                                 % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                 % end of if (1.b)
       end;                                     % end of if (1.a)
   end;                                         % end of elseif (1)
   %
   k = k+1;
end;                                            % end of loop for omega_1 %



for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                        % omega_new is  in RPM
```

```
omega_old = -100;                      % omega_old is in RPM
flg_slope = 1;
%
% determination of odd or even mode shapes
R = rem(count,2);
if R ==1
slope = 1;
elseif R == 0;
slope = -1;
else
display('This is an incorrect mode shape input!')
end % end of 'if loop R'
%-----------------------------------------------
% initial values
flag = 1;
flag_1 = 1;
det_bcold = 0;
det_bcnew = 0;
%------------------------------------------------
%
while flag == 1,
    F = eye(4);
    if omega_0 < 0,
        omega_0 = 0;
     % if real part of natural frequency is negative value make it zero
    end                    % end of if loop omega
    %--------------------------------------------
    for j = 1:n-1,
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
        G_n=[1,     0,        0,        -m_n(j+1)*omega_0^2   ;           % omega_0
is in radians
             0,     1,        0,        -T_n(j)           ;
             0,     0,        1,         0                  ;
             0,     0,        l_sn,      1                  ];
        %
        A_n=[1,             0,               0,                            0       ;
```

```
       l_sn,           1,               0,                           -T_n(j);
     -(l_sn^2)/(2*EI),   -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
   -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),    (T_n(j)*l_sn^3)/(3*EI),     1       ];
     %
     F_n = inv(G_n)*A_n;
     F = F_n * F;
 end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
 %
 B_c = [ F(2,3), F(2,4) ; F(4,3), F(4,4) ];
 det_bc = det(B_c);
 %----------------------------------------------------------
 if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),          % if loop 1
     omega_n(count) = omega_0*30/pi;     omega_0 is in radians, omega_n is in RPM
     %
     %--------------------------------
     if abs(omega_old-omega_new)<1e-10,                        % if loop 1-a
         disp('Warning!!  This value of omega may be in error.')
     end                                              % end of 'if loop 1-a'
     %-------------------------------
     % change the initial values
     flag = 0;
     flag_1 = 1;
     det_bcold = 0;
     det_bcnew = 0;
 else                                                  % else for if loop 1
     %-------------------------------
     if flag_1 == 1,                                       % if loop 1-b
         if det_bc > 0,                                    % if loop 1-b-i
             omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
         else                                        % else for if loop 1-b-i
             omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
         end;                                            % end of 'loop if 1-b-i'
         % rearrange the values
         omega_old = omega_0;
         omega_0 = omega_new;
         det_bcold = det_bc;
         flag_1 = 2;
     elseif flag_1 == 2,                         % elseif for if loop 1-b
         if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,     % if loop
1-b-ii
```

```
                  if det_bc > 0,                           % if loop 1-b-ii-a
                      omega_new  =  omega_0  -  (200*pi/30)*slope;  %  omega_new  is  in
radians
                  else                                % else for if loop 1-b-ii-a
                      omega_new  =  omega_0  +  (200*pi/30)*slope;  %  omega_new  is  in
radians
                  end;                                % end of 'if loop 1-b-ii-a
              % rearrange the values
              omega_old = omega_0;
              omega_0 = omega_new;
              det_bcold = det_bc;
              else                                    % else for if loop 1-b-ii
                  omega_0 = (omega_new + omega_old)/2;
                                          % omega_old,new and 0 are in radians
                  det_bcnew = det_bc;
                  flag_1 = 0;
              end                                         % end of 'if loop 1-b-ii'
          else                                        % else for if loop 1-b
              if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
                                                      % if loop 1-b-iii
                  omega_new = omega_0;
                  det_bcnew = det_bc;
              elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)), %
elseif for if loop 1-b-ii
                  omega_old = omega_0;
                  det_bcold = det_bc;
              end;                        % end of 'if loop 1-b-ii'
              omega_0 = (omega_new + omega_old)/2;
          end;                                        % end of 'if loop 1-b'
          %--------------------------------------
      end;                                            % end of 'if loop 1'
      %-------------------------------------------------------------------------------
  end;                                                % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% ************************************************************************************
%
for m= 1 : count                                               % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
  F = eye(4);
  for j = 1:n-1,                                          % for loop j
      %
      l_sn = r_n(j) - r_n(j+1);
      %
      EI = E_n * I_x(j);
      %
      G_n=[1,     0,        0,   -m_n(j+1)*(omega_n(m)*pi/30)^2;
          0,     1,        0,         -T_n(j);
          0,     0,        1,              0;
          0,     0,       l_sn,             1];

      A_n=[1,                 0,                  0,                 0;
           l_sn,              1,                  0,            -T_n(j);
          -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
          -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

      F_n = inv(G_n)*A_n;
      F = F_n * F;
      X_n(:,j+1) = F * X_tip;
  end;                     % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
  %
  % get the total deflection at each radial station
  deflection(m,:)=X_n(4,:);
  plot ( r_n, 2*deflection(m,:), '-');
  grid on;
  title  (  ' Flatwise   Mode  Shapes  at  Operational  Rotational  Velocity  for  Uniform
Blade' );
  xlabel ( ' Blade Station (inc)' ) ;
  ylabel ( ' Relative Deflection ' ) ;
  hold on
end % end of for loop count
hold off
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency to Rotational Speed is: ')
disp(omega_n/R_V)
omega_rate = omega_n/R_V;
%
```

```
% end of program "HINGED_UNIFORM_FLATWISE"
```

4. Hinged Uniform Edgewise Function

```
function    [omega_n,omega_rate,cent_force,varargout]    =    hinged_uniform_edgewise(E_n,
modeshp, R_V, e, R, weight,I_xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           This function if for the uniform hinged blades                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
segment     = R/50;
segment_W   = weight/50;
z           = 1;
r_n(1)      = R;
r_n(51)     = e;
I_x(51)     = 0.0001;
Weight_n(51)= 0.0001;
I_x(1)      = I_xx;
Weight_n(1) = segment_W;
%
for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
        if r_n(seg) < e
        r_n(seg) = e;
        end
    I_x(seg)       = I_xx;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;
%
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
R_V1= R_V * pi/30;                               % R_V1 is in radians
T_n(1)  = 0;                                       % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;
```

```
end; % end of loop for i--> { T_n(i) }
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%**********************************************************************************
k = 1;
i = 1;
for omega1 = 0 :modeshp^2:2.1*modeshp^2*200,      % omega1 is in RPM
    omega(k) = omega1*2*pi/60;          % omega(k) is in radians
    F = eye(4);
    %-------------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);



        EI = E_n * I_x(j);



        G_n=[1,     0,         0,           -m_n(j+1)*(omega(k)^2 + R_V1^2);          %
omega(k) is in radians
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];

        A_n=[1,                 0,                  0,                 0;
             l_sn,              1,                  0,             -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %--------------------------------------------------------------------------

    B_c = [ F(2,3),F(2,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);



    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
```

```
        omega_natural(i) = omega(k);
        i = i+1;
     else
        if k >1,                                  % if loop (1.a)
        s = k-1;
            if det_bc(k) * det_bc(:,s) < 0        % if loop (1.b)
            omega_natural(i) = (omega(k) + omega(:,s))/2;
            i = i +1;
            end;                                  % end of if (1.b)
        end;                                      % end of if (1.a)
    end;                                          % end of elseif (1)
    %
    k = k+1;
end;                                              % end of loop for omega_1
%



for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                        % omega_new is  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    % determination of odd or even mode shapes
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end % end of 'if loop R'
    %----------------------------------------------
    % initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %------------------------------------------------
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
            omega_0 = 0;          % if real part of natural frequency is negative value
make it zero
        end                     % end of if loop omega
        %------------------------------------------
        for j = 1:n-1,
            %
            l_sn = r_n(j) - r_n(j+1);
            %
            EI = E_n * I_x(j);
            %
             G_n=[1,     0,        0,        -m_n(j+1)*(omega_0^2 + R_V1^2)     ;
                 0,     1,        0,        -T_n(j)                 ;
                 0,     0,        1,         0                      ;
                 0,     0,        l_sn,      1                      ];
            %
            A_n=[1,                0,              0,                          0       ;
                 l_sn,            1,              0,                         -T_n(j);
           -(l_sn^2)/(2*EI),   -l_sn/EI,         1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
           -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),    (T_n(j)*l_sn^3)/(3*EI),     1     ];
            %
            F_n = inv(G_n)*A_n;
            F = F_n * F;
        end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
        %
        B_c = [ F(2,3), F(2,4) ; F(4,3), F(4,4) ];
        det_bc = det(B_c);
        %--------------------------------------------------------
        if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),             % if loop 1
            omega_n(count) = omega_0*30/pi;   % omega_0 is in radians, omega_n is in RPM
            %
            %--------------------------------
            if abs(omega_old-omega_new)<1e-10,                         % if loop 1-a
                disp('Warning!!  This value of omega may be in error.')
            end                                                % end of 'if loop 1-a'
            %-------------------------------
            % change the initial values
            flag = 0;
            flag_1 = 1;
            det_bcold = 0;
            det_bcnew = 0;
        else                                                    % else for if loop 1
            %-------------------------------
            if flag_1 == 1,                                            % if loop 1-b
                if det_bc > 0,                                     % if loop 1-b-i
                    omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
                else                                        % else for if loop 1-b-i
                    omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
                end;                                    % end of 'loop if 1-b-i'
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                flag_1 = 2;
            elseif flag_1 == 2,                         % elseif for if loop 1-b
                if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                                              % if loop 1-b-ii
                    if det_bc > 0,                                 % if loop 1-b-ii-a
                        omega_new = omega_0 - (200*pi/30)*slope;%omega_new is in radians
                    else                                    % else for if loop 1-b-ii-a
                        omega_new = omega_0 + (200*pi/30)*slope;%omega_new is in radians
                    end;                                  % end of 'if loop 1-b-ii-a
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                else                                         % else for if loop 1-b-ii
                    omega_0 = (omega_new + omega_old)/2; % omega_old,new and 0 are in
radians
                    det_bcnew = det_bc;
                    flag_1 = 0;
                end                                       % end of 'if loop 1-b-ii'
            else                                          % else for if loop 1-b
```

```
                if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
                                                        % if loop 1-b-iii
                    omega_new = omega_0;
                    det_bcnew = det_bc;
                elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),
                                                        % elseif for if loop 1-b-ii
                    omega_old = omega_0;
                    det_bcold = det_bc;
                end;                                      % end of 'if loop 1-b-ii'
                omega_0 = (omega_new + omega_old)/2;
            end;                                         % end of 'if loop 1-b'
            %----------------------------------------
        end;                                              % end of 'if loop 1'
        %--------------------------------------------------------------------------------
    end;                                         % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% ***********************************************************************************
%
for m= 1 : count                                        % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                                        % for loop j
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
        G_n=[1,     0,        0,   -m_n(j+1)*((omega_n(m)*pi/30)^2  + R_V1^2);
            0,     1,        0,         -T_n(j);
            0,     0,        1,              0;
            0,     0,       l_sn,             1];

        A_n=[1,                0,                  0,                 0;
             l_sn,             1,                  0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];
        F_n = inv(G_n)*A_n;
        F = F_n * F;
        X_n(:,j+1) = F * X_tip;
    end;                       %end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    plot ( r_n, 2*deflection(m,:), '-');
    grid on;
    title  (  '  Chordwise  Mode  Shapes  at  Operational  Rotational  Velocity  for  Uniform
Blade' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
%
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency to Rotational Speed is: ')
disp(omega_n/R_V)
omega_rate = omega_n/R_V;
%
               % end of function "HINGED_UNIFORM_EDGEWISE"
```

5. Hingeless Uniform Flatwise Function

```
function   [omega_n,omega_rate,cent_force,varargout]   =   hingeless_uniform_flatwise(E_n,
modeshp, R_V, e, R, weight,I_xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           This function if for the uniform hingeless blades                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
segment     = R/50;
segment_W   = weight/50;
z           = 1;
r_n(1)      = R;
r_n(51)     = 0;
I_x(51)     = 0.0001;
Weight_n(51)= 0.0001;
I_x(1)      = I_xx;
Weight_n(1) = segment_W;
```

```
e = 0;

for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
        if r_n(seg) < 0
        r_n(seg) = e;
        end
    I_x(seg)       = I_xx;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;
%
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
R_V1= R_V * pi/30;                              % R_V1 is in radians
T_n(1)  = 0;                                     % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;

end; % end of loop for i--> { T_n(i) }
%
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%*************************************************************************************
k = 1;
i = 1;
for omega1 = 0 :modeshp^2:2.1*modeshp^2*200,      % omega1 is in RPM
    omega(k) = omega1*2*pi/60;          % omega(k) is in radians
    F = eye(4);
    %--------------------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);



        EI = E_n * I_x(j);
        G_n=[1,     0,        0,          -m_n(j+1)*omega(k)^2; % omega(k) is in radians
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];

        A_n=[1,                0,                  0,                  0;
             l_sn,             1,                  0,              -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %---------------------------------------------------------------------------------


    B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);



    %determination of the points and the natural frequencies where det crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
    else
       if k >1,                                % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0      % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                % end of if (1.b)
       end;                                    % end of if (1.a)
   end;                                        % end of elseif (1)
   %
   k = k+1;
end;                                           % end of loop for omega_1 %
```

```
for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                        % omega_new is  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    % determination of odd or even mode shapes
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end % end of 'if loop R'
    %-----------------------------------------------
    % initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %------------------------------------------------
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
            omega_0 = 0;
         % if real part of natural frequency is negative value make it zero
        end                     % end of if loop omega
        %-------------------------------------------
        for j = 1:n-1,
            %
            l_sn = r_n(j) - r_n(j+1);
            %
            EI = E_n * I_x(j);
            %
            G_n=[1,     0,         0,        -m_n(j+1)*omega_0^2   ;
                                                         % omega_0 is in radians
                 0,     1,         0,        -T_n(j)           ;
                 0,     0,         1,         0                  ;
                 0,     0,         l_sn,      1                  ];
            %
            A_n=[1,                0,                 0,                       0       ;
             l_sn,           1,                0,                          -T_n(j);
           -(l_sn^2)/(2*EI),   -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
           -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),   (T_n(j)*l_sn^3)/(3*EI),     1       ];
            %
            F_n = inv(G_n)*A_n;
            F = F_n * F;
        end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
        %
        B_c = [ F(3,3), F(3,4) ; F(4,3), F(4,4) ];
        det_bc = det(B_c);
        %----------------------------------------------------------
        if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),         % if loop 1
            omega_n(count) = omega_0*30/pi;  % omega_0 is in radians, omega_n is in RPM
            %
            %--------------------------------
            if abs(omega_old-omega_new)<1e-10,                         % if loop 1-a
                disp('Warning!!  This value of omega may be in error.')
            end                                         % end of 'if loop 1-a'
            %-------------------------------
            % change the initial values
            flag = 0;
            flag_1 = 1;
            det_bcold = 0;
            det_bcnew = 0;
        else                                          % else for if loop 1
            %-------------------------------
            if flag_1 == 1,                                  % if loop 1-b
                if det_bc > 0,                              % if loop 1-b-i
                    omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
                else                                    % else for if loop 1-b-i
                    omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
                end;                                  % end of 'loop if 1-b-i'
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                flag_1 = 2;
            elseif flag_1 == 2,                         % elseif for if loop 1-b
                if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                                       % if loop 1-b-ii
                    if det_bc > 0,                           % if loop 1-b-ii-a
                        omega_new = omega_0 - (200*pi/30)*slope;
                                                % omega_new is in radians
                    else                               % else for if loop 1-b-ii-a
                        omega_new = omega_0 + (200*pi/30)*slope;
                                                % omega_new is in radians
                    end;                           % end of 'if loop 1-b-ii-a
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                else                                     % else for if loop 1-b-ii
                    omega_0 = (omega_new + omega_old)/2;
                                         % omega_old,new and 0 are in radians
                    det_bcnew = det_bc;
                    flag_1 = 0;
                end                                    % end of 'if loop 1-b-ii'
            else                                        % else for if loop 1-b
                if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
                                                              % if loop 1-b-iii
                    omega_new = omega_0;
                    det_bcnew = det_bc;
                elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),
                                                % elseif for if loop 1-b-ii
                    omega_old = omega_0;
                    det_bcold = det_bc;
                end;                                          % end of 'if loop 1-b-ii'
                omega_0 = (omega_new + omega_old)/2;
            end;                                          % end of 'if loop 1-b'
            %---------------------------------------
        end;                                             % end of 'if loop 1'
        %-------------------------------------------------------------------------------
```

```
    end;                                              % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% ************************************************************************************
%
for m= 1 : count                                    % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                      % for loop j
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
        G_n=[1,     0,        0,   -m_n(j+1)*(omega_n(m)*pi/30)^2;
            0,     1,        0,         -T_n(j);
            0,     0,        1,              0;
            0,     0,       l_sn,             1];

        A_n=[1,                0,                  0,                  0;
             l_sn,             1,                  0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
        X_n(:,j+1) = F * X_tip;
    end;                        % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    plot ( r_n, 2*deflection(m,:), '-');
    grid on
    title  (  '  Flatwise   Mode  Shapes  at  Operational  Rotational  Velocity  for  Uniform
Blade' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
```

```
end % end of for loop count
hold off
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency to Rotational Speed is: ')
disp(omega_n/R_V)
%
omega_rate = omega_n/R_V;
%
% end of program "Hingeless Uniform Flatwise"
```

6. Hingeless Uniform Edgewise Function

```
function  [omega_n,omega_rate,cent_force,varargout]  =  hingeless_uniform_edgewise(E_n,
modeshp, R_V, e, R, weight,I_xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           This function if for the uniform hingeless blades                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
segment     = R/50;
segment_W   = weight/50;
z           = 1;
r_n(1)      = R;
r_n(51)     = 0;
I_x(51)     = 0.0001;
Weight_n(51)= 0.0001;
I_x(1)      = I_xx;
Weight_n(1) = segment_W;

for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
        if r_n(seg) < 0
        r_n(seg) = 0;
        end
    I_x(seg)       = I_xx;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;

%
```

```
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
R_V1= R_V * pi/30;                                    % R_V1 is in radians
T_n(1)  = 0;                                           % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;

end; % end of loop for i--> { T_n(i) }
%
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%*******************************************************************************
k = 1;
i = 1;
for omega1 = 0 :modeshp^2:2.1*modeshp^2*200,      % omega1 is in RPM
    omega(k) = omega1*2*pi/60;          % omega(k) is in radians
    F = eye(4);
    %-----------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);


        EI = E_n * I_x(j);





        G_n=[1,      0,          0,           -m_n(j+1)*(omega(k)^2 + R_V1^2);          %
omega(k) is in radians
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];

        A_n=[1,                0,                 0,                 0;
             l_sn,             1,                 0,            -T_n(j);
```

```
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %---------------------------------------------------------------------------


    B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);



    %determination of the points and the natural frequencies where det crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
    else
       if k >1,                                 % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                 % end of if (1.b)
       end;                                     % end of if (1.a)
   end;                                         % end of elseif (1)
   %
   k = k+1;
end;                                            % end of loop for omega_1
%

%

for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
```

```
omega_new = 100;                        % omega_new is  in RPM
omega_old = -100;                       % omega_old is in RPM
flg_slope = 1;
%
% determination of odd or even mode shapes
R = rem(count,2);
if R ==1
slope = 1;
elseif R == 0;
slope = -1;
else
display('This is an incorrect mode shape input!')
end % end of 'if loop R'
%-----------------------------------------------
% initial values
flag = 1;
flag_1 = 1;
det_bcold = 0;
det_bcnew = 0;
%------------------------------------------------
%
while flag == 1,
    F = eye(4);
    if omega_0 < 0,
        omega_0 = 0;
     % if real part of natural frequency is negative value make it zero
    end                     % end of if loop omega
    %-------------------------------------------
    for j = 1:n-1,
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
         G_n=[1,     0,         0,         -m_n(j+1)*(omega_0^2 + R_V1^2)     ;
            0,     1,         0,         -T_n(j)                     ;
            0,     0,         1,          0                          ;
            0,     0,         l_sn,       1                          ];
        %
        A_n=[1,               0,                0,                         0       ;
         l_sn,           1,                  0,                             -T_n(j);
        -(l_sn^2)/(2*EI),   -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
       -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),   (T_n(j)*l_sn^3)/(3*EI),     1       ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    B_c = [ F(3,3), F(3,4) ; F(4,3), F(4,4) ];
    det_bc = det(B_c);
    %----------------------------------------------------------
    if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),        % if loop 1
        omega_n(count) = omega_0*30/pi;  omega_0 is in radians, omega_n is in RPM  %
        %--------------------------------
        if abs(omega_old-omega_new)<1e-10,                       % if loop 1-a
            disp('Warning!!  This value of omega may be in error.')
        end                                  % end of 'if loop 1-a'
        %-------------------------------
        % change the initial values
        flag = 0;
        flag_1 = 1;
        det_bcold = 0;
        det_bcnew = 0;
    else                                         % else for if loop 1
        %-------------------------------
        if flag_1 == 1,                            % if loop 1-b
            if det_bc > 0,                              % if loop 1-b-i
                omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
            else                                   % else for if loop 1-b-i
                omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
            end;                                      % end of 'loop if 1-b-i'
            % rearrange the values
            omega_old = omega_0;
            omega_0 = omega_new;
            det_bcold = det_bc;
            flag_1 = 2;
        elseif flag_1 == 2,                    % elseif for if loop 1-b
            if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                                 % if loop 1-b-ii
                if det_bc > 0,                        % if loop 1-b-ii-a
```

```
                        omega_new = omega_0 - (200*pi/30)*slope;
                                   % omega_new is in radians
                    else                               % else for if loop 1-b-ii-a
                        omega_new  =  omega_0  +  (200*pi/30)*slope;  %  omega_new  is  in
radians
                    end;                          % end of 'if loop 1-b-ii-a
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                else                                   % else for if loop 1-b-ii
                    omega_0  =  (omega_new  +  omega_old)/2;  %  omega_old,new  and  0  are  in
radians
                    det_bcnew = det_bc;
                    flag_1 = 0;
                end                                  % end of 'if loop 1-b-ii'
            else                                    % else for if loop 1-b
                if  ((det_bc>0  &  det_bcnew  >  0)  |  (det_bc<0  &  det_bcnew  <  0)),    %  if
loop 1-b-iii
                    omega_new = omega_0;
                    det_bcnew = det_bc;
                elseif  ((det_bc  <  0  &  det_bcnew  >  0)  |  (det_bc  >  0  &  det_bcnew  <  0)),  %
elseif for if loop 1-b-ii
                    omega_old = omega_0;
                    det_bcold = det_bc;
                end;                                   % end of 'if loop 1-b-ii'
                omega_0 = (omega_new + omega_old)/2;
            end;                                       % end of 'if loop 1-b'
            %--------------------------------------
        end;                                             % end of 'if loop 1'
        %---------------------------------------------------------------------
    end;                                         % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% ****************************************************************************
%
for m= 1 : count                                     % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                    % for loop j
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
        G_n=[1,     0,        0,   -m_n(j+1)*((omega_n(m)*pi/30)^2  + R_V1^2);
            0,     1,        0,         -T_n(j);
            0,     0,        1,              0;
            0,     0,       l_sn,             1];

        A_n=[1,                 0,                   0,                  0;
             l_sn,              1,                   0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
        X_n(:,j+1) = F * X_tip;
    end;                            % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    plot ( r_n, 2*deflection(m,:), '-');
    grid on;
    title  (  '  Chordwise  Mode  Shapes  at  Operational  Rotational  Velocity  for  Uniform
Blade' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
%
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency to Rotational Speed is: ')
disp(omega_n/R_V)
omega_rate = omega_n/R_V;
%
% end of function "HINGELESS_UNIFORM_EDGEWISE"
```

7. Hinged Nonuniform Flatwise Function

```
function   [omega_n,omega_rate,cent_force,varargout]   =   hingeless_uniform_edgewise(E_n, modeshp, R_V, e, R, weight,I_xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           This function if for the uniform hingeless blades                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
segment     = R/50;
segment_W   = weight/50;
z           = 1;
r_n(1)      = R;
r_n(51)     = 0;
I_x(51)     = 0.0001;
Weight_n(51)= 0.0001;
I_x(1)      = I_xx;
Weight_n(1) = segment_W;

for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
        if r_n(seg) < 0
        r_n(seg) = 0;
        end
    I_x(seg)       = I_xx;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;

%
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
R_V1= R_V * pi/30;                             % R_V1 is in radians
T_n(1)  = 0;                                     % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;

end; % end of loop for i--> { T_n(i) }
```

```
%
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%***************************************************************************
k = 1;
i = 1;
for omega1 = 0 :modeshp^2:2.1*modeshp^2*200,      % omega1 is in RPM
    omega(k) = omega1*2*pi/60;          % omega(k) is in radians
    F = eye(4);
    %-------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);



        EI = E_n * I_x(j);





        G_n=[1,      0,          0,            -m_n(j+1)*(omega(k)^2 + R_V1^2);          %
omega(k) is in radians
             0,     1,         0,           -T_n(j)         ;
             0,     0,         1,            0                ;
             0,     0,         l_sn,         1                ];

        A_n=[1,                 0,                  0,                  0;
             l_sn,              1,                  0,            -T_n(j);
            -(l_sn^2)/(2*EI),      -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %-------------------------------------------------------------------------

    B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);
    %determination of the points and the natural frequencies where det crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
    else
       if k >1,                                % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                % end of if (1.b)
       end;                                    % end of if (1.a)
   end;                                        % end of elseif (1)
   %
   k = k+1;
end;                                           % end of loop for omega_1
%

%

for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                        % omega_new is  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    % determination of odd or even mode shapes
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
```

```
display('This is an incorrect mode shape input!')
end % end of 'if loop R'
%----------------------------------------------
% initial values
flag = 1;
flag_1 = 1;
det_bcold = 0;
det_bcnew = 0;
%-----------------------------------------------
%
while flag == 1,
    F = eye(4);
    if omega_0 < 0,
        omega_0 = 0;
     % if real part of natural frequency is negative value make it zero
    end                     % end of if loop omega
    %-------------------------------------------
    for j = 1:n-1,
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
         G_n=[1,     0,        0,        -m_n(j+1)*(omega_0^2 + R_V1^2)     ;
             0,     1,        0,        -T_n(j)                  ;
             0,     0,        1,         0                       ;
             0,     0,        l_sn,      1                       ];
        %
        A_n=[1,              0,                0,                         0       ;
         l_sn,           1,                  0,                            -T_n(j);
        -(l_sn^2)/(2*EI),   -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
       -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),   (T_n(j)*l_sn^3)/(3*EI),     1       ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    B_c = [ F(3,3), F(3,4) ; F(4,3), F(4,4) ];
    det_bc = det(B_c);
    %---------------------------------------------------------
```

```
        if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),        % if loop 1
            omega_n(count) = omega_0*30/pi;  omega_0 is in radians, omega_n is in RPM  %
            %--------------------------------
            if abs(omega_old-omega_new)<1e-10,                        % if loop 1-a
                disp('Warning!!  This value of omega may be in error.')
            end                                    % end of 'if loop 1-a'
            %-------------------------------
            % change the initial values
            flag = 0;
            flag_1 = 1;
            det_bcold = 0;
            det_bcnew = 0;
        else                                            % else for if loop 1
            %-------------------------------
            if flag_1 == 1,                               % if loop 1-b
                if det_bc > 0,                                % if loop 1-b-i
                    omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
                else                                      % else for if loop 1-b-i
                    omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
                end;                                         % end of 'loop if 1-b-i'
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                flag_1 = 2;
            elseif flag_1 == 2,                       % elseif for if loop 1-b
                if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                                        % if loop 1-b-ii
                    if det_bc > 0,                          % if loop 1-b-ii-a
                        omega_new = omega_0 - (200*pi/30)*slope;
                                   % omega_new is in radians
                    else                                % else for if loop 1-b-ii-a
                        omega_new  =  omega_0  +  (200*pi/30)*slope;  %  omega_new  is  in
radians
                    end;                           % end of 'if loop 1-b-ii-a
                % rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                else                                    % else for if loop 1-b-ii
```

```
                    omega_0 = (omega_new + omega_old)/2; % omega_old,new and 0 are in
radians
                    det_bcnew = det_bc;
                    flag_1 = 0;
                end                                  % end of 'if loop 1-b-ii'
            else                                     % else for if loop 1-b
                if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),    % if
loop 1-b-iii
                    omega_new = omega_0;
                    det_bcnew = det_bc;
                elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)), %
elseif for if loop 1-b-ii
                    omega_old = omega_0;
                    det_bcold = det_bc;
                end;                                     % end of 'if loop 1-b-ii'
                omega_0 = (omega_new + omega_old)/2;
            end;                                         % end of 'if loop 1-b'
            %----------------------------------------
        end;                                               % end of 'if loop 1'
        %--------------------------------------------------------------------
    end;                                           % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% ********************************************************************************
%
for m= 1 : count                                     % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                     % for loop j
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
        G_n=[1,     0,        0,   -m_n(j+1)*((omega_n(m)*pi/30)^2  + R_V1^2);
            0,     1,        0,         -T_n(j);
            0,     0,        1,              0;
            0,     0,       l_sn,             1];
        A_n=[1,                0,                  0,                 0;
             l_sn,             1,                  0,             -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
        X_n(:,j+1) = F * X_tip;
    end;                             % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    plot ( r_n, 2*deflection(m,:), '-');
    grid on;
    title  (  '  Chordwise  Mode  Shapes  at  Operational  Rotational  Velocity  for  Uniform
Blade' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
%
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency to Rotational Speed is: ')
disp(omega_n/R_V)
omega_rate = omega_n/R_V;
%
% end of function "HINGELESS_UNIFORM_EDGEWISE"
```

8. Hinged Nonuniform Edgewise Function

```
function  [omega_n,  omega_rate,cent_force,varargout]  =  hinged_nonuniform_edgewise(E_n,
modeshp, R_V, e, R, weight,I_xx)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           This function is for                                   %
%                          Hinged NonUniform Blades                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
R_n     = R;
```

```
%
W_n     = weight;
%
m_n     = W_n/(32.174*12);
%
n       = length(R_n);
%
I_yy = I_xx;
%
if R_n(n) < e
    R_n(n) = e;
end
R_n(n) = e;
%
%==============================================================================
%
omega   = 0;
det_Bc  = 0;
%
R_V1    = R_V * pi/30;                              % radians (R_V1)
%
%=============================================================================
% Calculation of "Centrifugal Force" at the root:
%-----------------------------------------------------------------------------
T_n(1)  = m_n(1) * R_n(1) * R_V1^2;             % (T_n) and radians (R_V1)
%
% T_n(1) is is the "Centrifugal Force at the TIP"
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * R_n(i+1) * R_V1^2;
end;                                           % end of loop for i--> { T_n(i) }
%
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
% Displays "Centrifugal Force at r "
%
%=============================================================================
k       = 1;
i       = 1;
```

```
%
for omega_1 = 0 : modeshp^2*R_V/500 : 2.1*modeshp^2*R_V,
                                               % RPM (omega_1)
    omega(k) = omega_1*2*pi/60;                % radians (omega(k))
    F = eye(4);
    %----------------------------------------------------------------------------
    for j = 1:n-1,
        % determine length of the segments;
        l_sn = R_n(j) - R_n(j+1);
        %
        % detemine the stiffness
        EI = E_n * I_yy(j);
        %
        G_n = [ 1,     0,        0,          -m_n(j+1)*(omega(k)^2 + R_V1^2);
                0,     1,        0,          -T_n(j)              ;
                0,     0,        1,           0                   ;
                0,     0,        l_sn,        1                  ];

        A_n = [ 1,                 0,                 0,                         0      ;
                l_sn,              1,                 0,                       -T_n(j) ;
               -(l_sn^2)/(2*EI),  -l_sn/EI,           1+(T_n(j)*l_sn^2)/(2*EI), 0      ;
               -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),   (T_n(j)*l_sn^3)/(3*EI),   1     ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end;                         % end of loop for j --> { A_n, G_n, F_n )
    %
    %-------------------------------------------------------------------------
    B_c = [ F(2,3),F(2,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);
    %
    %----------------------------------------------------------------------
    %determination of the points and the natural frequencies where det crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
   else
       if k >1,                                % if loop (1.a)
```

```
        s = k-1;
            if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
            omega_natural(i) = (omega(k) + omega(:,s))/2;
            i = i +1;
            end;                                 % end of if (1.b)
        end;                                     % end of if (1.a)
    end;                                         % end of elseif (1)
    %
    k = k+1;
end;                                             % end of loop for omega_1
%
%===============================================================================
%
% Sorting and Displaying the "Natural Frequencies"
%
% Sort and convert the unit of natural frequency to 'cpm'
for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;                                             % end of loop for (m)
%

%==============================================================================
%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % radians (omega_0) ,RPM (omega_n)
    %
    omega_new = 100;                        % omega_new i  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    %-----------------------------------------------------------------------
    % Determination of odd or even mode shapes
    %
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end                                     % end of 'if loop R'
    %
    %-----------------------------------------------------------------------
    % Initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %
    %--------------------------------------------------------------------
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
           omega_0 = 0;                      % if natural frequency is negative, make it
zero
        end                                 % end of if loop omega
        %
        for j = 1:n-1,
            %
            l_sn = R_n(j) - R_n(j+1);
            %
            EI = E_n * I_yy(j);
            %
            G_n=[1,     0,        0,        -m_n(j+1)*(omega_0^2 + R_V1^2)     ;
                 0,     1,        0,        -T_n(j)                 ;
                 0,     0,        1,         0                      ;
                 0,     0,        l_sn,      1                      ];
            %
            A_n=[1,              0,                0,                         0       ;
                 l_sn,           1,                0,                       -T_n(j)   ;
            -(l_sn^2)/(2*EI),  -l_sn/EI,       1+(T_n(j)*l_sn^2)/(2*EI),      0       ;
            -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),    (T_n(j)*l_sn^3)/(3*EI),   1       ];
            %
            F_n = inv(G_n)*A_n;
            F = F_n * F;
        end;                          % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
        %
        B_c = [ F(2,3), F(2,4) ; F(4,3), F(4,4) ];
        det_bc = det(B_c);
```

```
%--------------------------------------------------------------------------------
%
if (abs(det_bc)<1000 | abs(omega_old-omega_new)<1e-10),
                                    % if loop 1
    omega_n(count) = omega_0*30/pi; % radians (omega_0), RPM (omega_n)
    %
    if abs(omega_old-omega_new)<1e-10,               % if loop 1.a
        disp('Warning!!  This value of omega may be in error.')
    end                                % end of 'if loop 1.a'
    %
    % change the initial values
    flag = 0;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %
else                                          % else for if loop 1
    %
    if flag_1 == 1,                                  % if loop 1.b
        if det_bc > 0,                                   % if loop 1.b.i
            omega_new = omega_0 - (200*pi/30)*slope;    % radians(omega_new)
        else                           % else for if loop 1.b.i
            omega_new = omega_0 + (200*pi/30)*slope;
        end;                                % end of 'loop if 1.b.i'
        %
        % Rearrange the values
        %
        omega_old = omega_0;
        omega_0 = omega_new;
        det_bcold = det_bc;
        flag_1 = 2;
        %
    elseif flag_1 == 2,                        % elseif for if loop 1.b
        if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                               % if loop 1.b.ii
            if det_bc > 0,                         % if loop 1.b.ii.a
                omega_new = omega_0 - (200*pi/30)*slope;
            else                               % else for if loop 1.b.ii.a
                omega_new = omega_0 + (200*pi/30)*slope;
            end;                                 % end of 'if loop 1.b.ii.a
```

```
                %
                % Rearrange the values
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;

            else                              % else for if loop 1.b.ii
                omega_0 = (omega_new + omega_old)/2;
                det_bcnew = det_bc;
                flag_1 = 0;
            end                              % end of 'if loop 1.b.ii'
        else                                  % else for if loop 1-b
            if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),% if loop
1.b.iii
                omega_new = omega_0;
                det_bcnew = det_bc;
            elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),%
elseif for if loop 1.b.ii
                omega_old = omega_0;
                det_bcold = det_bc;
            end;                             % end of 'if loop 1.b.ii'
            %
            omega_0 = (omega_new + omega_old)/2;
            %
        end;                                  % end of 'if loop 1.b'
        %
    end;                                      % end of 'if loop 1'
    %
  end;                                        % end of 'while loop'
 %
 theta_n(count) = -B_c(1,2) / B_c(1,1);
 % According to the equation we can also use as : theta_n(count)= -B_c(2,2)/B_c(2,1)
 %
end                                                                    % end of loop for
'count'
%
%================================================================================
for m= 1 : count                                   % for loop 'm'
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                          % for loop 'j'
        %
        l_sn    = R_n(j) - R_n(j+1);
        %
        EI      = E_n * I_yy(j);
        %
        G_n     =[ 1,      0,         0,       -m_n(j+1)*((omega_n(m)*pi/30)^2  + R_V1^2);
                   0,      1,         0,       -T_n(j)                        ;
                   0,      0,         1,        0                             ;
                   0,      0,       l_sn,             1                       ];
        %
        A_n     =[ 1,             0,               0,                       0       ;
                 l_sn,           1,               0,                     -T_n(j)  ;
               -(l_sn^2)/(2*EI),  -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),  0       ;
              -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),   1       ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
        %
        X_n(:,j+1) = F * X_tip;
        %
    end;                    % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % Get the total deflection at each radial station
    %
    deflection(m,:)=X_n(4,:);
    plot ( R_n, X_n(4,:), '-')


    %plot ( R_n, 2*X_n(4,:), '-');
    grid on;
    title ( ' Chordwise Mode Shapes at Operational Rotational Velocity ' );
    xlabel ( ' Blade Station (inc)' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end % end of for loop count
hold off
disp('The Natural Frequencies are(cpm): ')
disp(R_V)
%
```

```
disp('The ratio of the Natural Frequency/Rotational Speed is: ')
disp(omega_n/R_V)

omega_rate = omega_n/R_V;

%
% end of program "HINGED_NONUNIFORM_EDGEWISE"
```

9. Hingeless Nonuniform Flatwise Function

```
function  [omega_n,omega_rate,cent_force,varargout]  =  hingeless_nonuniform_flatwise(E_n,
modeshp, R_V, e, R, weight,I_xx)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           This function is for                                    %
%                          Hingeless NonUniform Blades                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R_n     = R;
%
W_n     = weight;
%
m_n     = W_n/(32.174*12);
%
n       = length(R_n);
%
e = 0;
if R_n(n) < e
    R_n(n) = e;
end
R_n(n) = e;
%
% n represents the number of the stations(segments);
%
%=====================================================================================
%
omega   = 0;
det_Bc  = 0;
%
R_V1    = R_V * pi/30;                              % radians (R_V1)
%
%====================================================================================
% Calculation of "Centrifugal Force" at the root:
%-------------------------------------------------------------------------------------
T_n(1)  = m_n(1) * R_n(1) * R_V1^2;              % ....... (T_n) and radians (R_V1)
%
% T_n(1) is is the "Centrifugal Force at the TIP"
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * R_n(i+1) * R_V1^2;
end;                                            % end of loop for i--> { T_n(i) }
%
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
% Displays "Centrifugal Force at the root"
%
%=====================================================================================
k       = 1;
i       = 1;
%
for omega_1 = 0 : modeshp^2*R_V/200 : 2.1*modeshp^2*R_V,
                                                % RPM (omega_1)
    omega(k) = omega_1*2*pi/60;                 % radians (omega(k))
    F = eye(4);
    %-------------------------------------------------------------------------------------
    for j = 1:n-1,
        % determine the length of the segments
        l_sn = R_n(j) - R_n(j+1);
        %
        % determine the stiffness
        EI = E_n * I_xx(j);
        %
        G_n=[1,     0,        0,          -m_n(j+1)*omega(k)^2; % omega(k) is in radians
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];
        %
        A_n=[1,                 0,                  0,                  0;
             l_sn,              1,                  0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];
        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %--------------------------------------------------------------------------
    B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);
     %
    %----------------------------------------------------------------------------
    % determination of the points and the natural frequencies where det. crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
   else                                           % elseif loop (1)
       if k >1,                                % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0       % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                % end of if (1.b)
       end;                                    % end of if (1.a)
   end;                                        % end of elseif (1)
   %
   k = k+1;
end;                                           % end of loop for omega_1
%
%===========================================================================
%
% Sorting and Displaying the "Natural Frequencies"
%
% Sort and convert the unit of Natural Frequency to 'cpm'
for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;                                           % end of loop for (m)
%

%=============================================================================
%
```

```
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % radians (omega_0) ,RPM (omega_n)
    %
    omega_new = 100;                        % omega_new i  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    %-----------------------------------------------------------------------
    % Determination of odd or even mode shapes
    %
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end                                     % end of 'if loop R'
    %
    %------------------------------------------------------------------------
    % Initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %
    %---------------------------------------------------------------------
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
           omega_0 = 0;                      % if natural frequency is negative, make it
zero
        end                                 % end of if loop omega
        %
        %-------------------------------------------------------------------
        for j = 1:n-1,
            %
            l_sn = R_n(j) - R_n(j+1);
            %
            EI = E_n * I_xx(j);
            %
            G_n=[1,     0,        0,        -m_n(j+1)*omega_0^2   ;
                 0,     1,        0,        -T_n(j)             ;
                 0,     0,        1,         0                   ;
                 0,     0,        l_sn,      1                   ];
            %
            A_n=[1,                0,              0,                          0       ;
              l_sn,             1,              0,                          -T_n(j);
            -(l_sn^2)/(2*EI),   -l_sn/EI,         1+(T_n(j)*l_sn^2)/(2*EI),   0       ;
           -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),  (T_n(j)*l_sn^3)/(3*EI),     1       ];
            %
            F_n = inv(G_n)*A_n;
            F = F_n * F;
        end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
        %
        B_c = [ F(3,3), F(3,4) ; F(4,3), F(4,4) ];
        det_bc = det(B_c);
        %----------------------------------------------------------------------------
  %-------------------------------------------------------------------------------------
        %
        if (abs(det_bc)<10000 | abs(omega_old-omega_new)<1e-10),
                                            % if loop 1
            omega_n(count) = omega_0*30/pi; % radians (omega_0), RPM (omega_n)
            %
            if abs(omega_old-omega_new)<1e-10,                   % if loop 1.a
                disp('Warning!!  This value of omega may be in error.')
            end                                                % end of 'if loop 1.a'
            %
            % change the initial values
            flag = 0;
            flag_1 = 1;
            det_bcold = 0;
            det_bcnew = 0;
            %
        else                                                    % else for if loop 1
            %
            if flag_1 == 1,                                     % if loop 1.b
                if det_bc > 0,                                  % if loop 1.b.i
                    omega_new = omega_0 - (200*pi/30)*slope;   % radians(omega_new)
                else                                        % else for if loop 1.b.i
                    omega_new = omega_0 + (200*pi/30)*slope;
                end;                                      % end of 'loop if 1.b.i'
                %
                % Rearrange the values
                %
                omega_old = omega_0;
                omega_0 = omega_new;
                det_bcold = det_bc;
                flag_1 = 2;
                %
            elseif flag_1 == 2,                          % elseif for if loop 1.b
                if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,  % if loop
1.b.ii
                    if det_bc > 0,                       % if loop 1.b.ii.a
                        omega_new = omega_0 - (200*pi/30)*slope;
                    else                             % else for if loop 1.b.ii.a
                        omega_new = omega_0 + (200*pi/30)*slope;
                    end;                              % end of 'if loop 1.b.ii.a
                    %
                    % Rearrange the values
                    omega_old = omega_0;
                    omega_0 = omega_new;
                    det_bcold = det_bc;


                else                                        % else for if loop 1.b.ii
                    omega_0 = (omega_new + omega_old)/2;
                    det_bcnew = det_bc;
                    flag_1 = 0;
                end                                       % end of 'if loop 1.b.ii'
            else                                          % else for if loop 1-b
                if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
                                                        % if loop 1.b.iii
                    omega_new = omega_0;
                    det_bcnew = det_bc;
                elseif ((det_bc < 0 & det_bcnew > 0) | (det_bc > 0 & det_bcnew < 0)),%
elseif for if loop 1.b.ii
                    omega_old = omega_0;
                    det_bcold = det_bc;
                end;                                % end of 'if loop 1.b.ii'
                %
```

```
                omega_0 = (omega_new + omega_old)/2;
                %
            end;                                       % end of 'if loop 1.b'
            %
        end;                                            % end of 'if loop 1'
        %
    end;                                               % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
 % According to the equation we can also use as : theta_n(count)= -B_c(2,2)/B_c(2,1)
 %
end                                                    % end of loop for 'count'
%
%===================================================================================
 %
%
for m= 1 : count                                           % for loop 'm'
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                         % for loop 'j'
        %
        l_sn    = R_n(j) - R_n(j+1);
        %
        EI      = E_n * I_xx(j);
        %
        G_n     =[ 1,     0,         0,      -m_n(j+1)*(omega_n(m)*pi/30)^2  ;
                   0,     1,         0,      -T_n(j)                         ;
                   0,     0,         1,       0                              ;
                   0,     0,       l_sn,      1                              ];
        %
        A_n     =[ 1,                 0,                0,                   0       ;
                   l_sn,            1,               0,                  -T_n(j)   ;
             -(l_sn^2)/(2*EI),  -l_sn/EI,         1+(T_n(j)*l_sn^2)/(2*EI),  0       ;
            -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),  (T_n(j)*l_sn^3)/(3*EI),     1     ];
        %
        F_n = inv(G_n)*A_n;
        F = F_n * F;
        %
        X_n(:,j+1) = F * X_tip;
        %
    end;                       % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % Get the total deflection at each radial station
    %
    deflection(m,:)=X_n(4,:);
    plot ( R_n , 2*X_n(4,:), '-')
    grid on;
    title   (  '  Flatwise  Mode  Shapes  Of  Hingeless(Rigid)  Rotor  Blade  for  Nonuniform
Blade' );
    xlabel ( ' Blade Station ' ) ;
    ylabel ( ' Relative Deflection ' ) ;
    hold on
end                                               % end of for loop count
hold off
disp('The Natural Frequencies are(cpm): ')
disp(omega_n)
%
disp('The ratio of the Natural Frequency/Rotational Speed is: ')
disp(omega_n/R_V)
%
omega_rate = omega_n/R_V;
%
% end of function "HINGELESS_NONUNIFORM_FLATWISE"
```

10. Hingeless Nonuniform Edgewise Function

```
    function [omega_n,omega_rate,cent_force,varargout] = hingeless_uniform_edgewise(E_n,
modeshp, R_V, e, R, weight,I_xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           This function if for the uniform hingeless blades                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
segment     = R/50;
segment_W   = weight/50;
z           = 1;
r_n(1)      = R;
r_n(51)     = 0;
I_x(51)     = 0.0001;
Weight_n(51)= 0.0001;
I_x(1)      = I_xx;
```

```
Weight_n(1) = segment_W;

for seg = 2:50
    r_n(seg)        = r_n(z) - segment;
        if r_n(seg) < 0
        r_n(seg) = 0;
        end
    I_x(seg)       = I_xx;
    Weight_n(seg)   = segment_W;
    z = z+1;
end;

%
n = length(r_n);
%
m_n = Weight_n/(32.174*12);
%
R_V1= R_V * pi/30;                              % R_V1 is in radians
T_n(1)  = 0;                                      % the unit is radians
%
for i = 1 : n-1,
    T_n(i+1)=T_n(i) + m_n(i+1) * r_n(i+1) * R_V1^2;

end; % end of loop for i--> { T_n(i) }
%
cent_force = T_n(n);
disp( ' Centrifugal Force : ' )
disp(T_n(n))
%
%*****************************************************************************
k = 1;
i = 1;
for omega1 = 0 :modeshp^2:2.1*modeshp^2*200,      % omega1 is in RPM
    omega(k) = omega1*2*pi/60;          % omega(k) is in radians
    F = eye(4);
    %-------------------------------------------------------------------------
    for j = 1:n-1,
        l_sn = r_n(j) - r_n(j+1);
        EI = E_n * I_x(j);




        G_n=[1,     0,        0,          -m_n(j+1)*(omega(k)^2 + R_V1^2);
                                                      % omega(k) is in radians
             0,     1,        0,          -T_n(j)         ;
             0,     0,        1,           0                ;
             0,     0,        l_sn,        1                ];

        A_n=[1,                 0,                  0,                  0;
             l_sn,              1,                  0,             -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
    end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
    %------------------------------------------------------------------------

    B_c = [ F(3,3),F(3,4) ; F(4,3),F(4,4)];
    det_bc(k) = det(B_c);



    %determination of the points and the natural frequencies where det crosses the "0
%line"
    %
    if    det_bc(k) < 0.0001 & det_bc(k) > -0.0001 % elseif loop (1)
       omega_natural(i) = omega(k);
       i = i+1;
    else
       if k >1,                                  % if loop (1.a)
       s = k-1;
           if det_bc(k) * det_bc(:,s) < 0        % if loop (1.b)
           omega_natural(i) = (omega(k) + omega(:,s))/2;
           i = i +1;
           end;                                  % end of if (1.b)
       end;                                      % end of if (1.a)
   end;                                          % end of elseif (1)
   %
   k = k+1;
end;                                             % end of loop for omega_1
%

%

for  m = 1:modeshp,
    omega_n(m) = omega_natural(m)*30/pi;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1 : modeshp
    omega_0 = real(omega_n(count)*pi/30);   % omega_0 is in radians , omega_n is in RPM
    omega_new = 100;                        % omega_new is  in RPM
    omega_old = -100;                       % omega_old is in RPM
    flg_slope = 1;
    %
    % determination of odd or even mode shapes
    R = rem(count,2);
    if R ==1
    slope = 1;
    elseif R == 0;
    slope = -1;
    else
    display('This is an incorrect mode shape input!')
    end % end of 'if loop R'
    %-----------------------------------------------
    % initial values
    flag = 1;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
    %------------------------------------------------
    %
    while flag == 1,
        F = eye(4);
        if omega_0 < 0,
            omega_0 = 0;
             % if real part of natural frequency is negative value make it zero
        end                     % end of if loop omega
```

```
%-------------------------------------------
for j = 1:n-1,
    %
    l_sn = r_n(j) - r_n(j+1);
    %
    EI = E_n * I_x(j);
    %
     G_n=[1,     0,        0,        -m_n(j+1)*(omega_0^2 + R_V1^2)     ;
         0,     1,        0,        -T_n(j)                  ;
         0,     0,        1,         0                       ;
         0,     0,        l_sn,      1                       ];
    %
    A_n=[1,            0,               0,                           0       ;
     l_sn,          1,               0,                          -T_n(j);
   -(l_sn^2)/(2*EI),   -l_sn/EI,        1+(T_n(j)*l_sn^2)/(2*EI),     0       ;
  -(l_sn^3)/(3*EI),  -(l_sn^2)/(2*EI),    (T_n(j)*l_sn^3)/(3*EI),    1       ];
    %
    F_n = inv(G_n)*A_n;
    F = F_n * F;
end; % end of loo for j --> { l_sn, EI, A_n, G_n, F_n)
%
B_c = [ F(3,3), F(3,4) ; F(4,3), F(4,4) ];
det_bc = det(B_c);
%---------------------------------------------------------
if (abs(det_bc)<100 | abs(omega_old-omega_new)<1e-10),         % if loop 1
    omega_n(count) = omega_0*30/pi;   % omega_0 is in radians, omega_n is in RPM
    %
    %---------------------------------
    if abs(omega_old-omega_new)<1e-10,                      % if loop 1-a
        disp('Warning!!  This value of omega may be in error.')
    end                                         % end of 'if loop 1-a'
    %--------------------------------
    % change the initial values
    flag = 0;
    flag_1 = 1;
    det_bcold = 0;
    det_bcnew = 0;
else                                                    % else for if loop 1
    %--------------------------------
    if flag_1 == 1,                                      % if loop 1-b
```

```
            if det_bc > 0,                               % if loop 1-b-i
                omega_new = omega_0 - (200*pi/30)*slope; % omega_new is in radians
            else                                       % else for if loop 1-b-i
                omega_new = omega_0 + (200*pi/30)*slope; % omega_new is in radians
            end;                                      % end of 'loop if 1-b-i'
            % rearrange the values
            omega_old = omega_0;
            omega_0 = omega_new;
            det_bcold = det_bc;
            flag_1 = 2;
        elseif flag_1 == 2,                      % elseif for if loop 1-b
            if abs(det_bcold + det_bc) > abs(det_bc) & det_bcnew == 0,
                                    % if loop 1-b-ii
                if det_bc > 0,                      % if loop 1-b-ii-a
                    omega_new  =  omega_0  -  (200*pi/30)*slope;  %  omega_new  is  in
radians
                else                             % else for if loop 1-b-ii-a
                    omega_new = omega_0 + (200*pi/30)*slope;
                                    % omega_new is in radians
                end;                    % end of 'if loop 1-b-ii-a
            % rearrange the values
            omega_old = omega_0;
            omega_0 = omega_new;
            det_bcold = det_bc;
            else                                   % else for if loop 1-b-ii
                omega_0  =  (omega_new  +  omega_old)/2;  %  omega_old,new  and  0  are  in
radians
                det_bcnew = det_bc;
                flag_1 = 0;
            end                                  % end of 'if loop 1-b-ii'
        else                                       % else for if loop 1-b
            if ((det_bc>0 & det_bcnew > 0) | (det_bc<0 & det_bcnew < 0)),
                                    % if loop 1-b-iii
                omega_new = omega_0;
                det_bcnew = det_bc;
            elseif  ((det_bc  <  0  &  det_bcnew  >  0)  |  (det_bc  >  0  &  det_bcnew  <  0)),  %
elseif for if loop 1-b-ii
                omega_old = omega_0;
                det_bcold = det_bc;
            end;                                      % end of 'if loop 1-b-ii'
            omega_0 = (omega_new + omega_old)/2;
            end;                                          % end of 'if loop 1-b'
            %---------------------------------------
        end;                                              % end of 'if loop 1'
        %-------------------------------------------------------------------------------
    end;                                              % end of 'while loop'
 %
 theta_n(count) = -B_c(2,2) / B_c(2,1);
end % end of loop for count
% *********************************************************************************
%
for m= 1 : count                                      % for loop count
    X_tip = [0 ; 0 ; theta_n(m) ; 1 ];
    X_n(:,1) = X_tip;
    F = eye(4);
    for j = 1:n-1,                                        % for loop j
        %
        l_sn = r_n(j) - r_n(j+1);
        %
        EI = E_n * I_x(j);
        %
        G_n=[1,     0,        0,   -m_n(j+1)*((omega_n(m)*pi/30)^2  + R_V1^2);
            0,     1,        0,         -T_n(j);
            0,     0,        1,              0;
            0,     0,       l_sn,             1];

        A_n=[1,                 0,                  0,                 0;
             l_sn,              1,                  0,            -T_n(j);
            -(l_sn^2)/(2*EI),     -l_sn/EI,    1+(T_n(j)*l_sn^2)/(2*EI),    0;
            -(l_sn^3)/(3*EI), -(l_sn^2)/(2*EI), (T_n(j)*l_sn^3)/(3*EI),     1];

        F_n = inv(G_n)*A_n;
        F = F_n * F;
        X_n(:,j+1) = F * X_tip;
    end;                         % end of loop for j --> { l_sn, EI, A_n, G_n, F_n)
    %
    % get the total deflection at each radial station
    deflection(m,:)=X_n(4,:);
    plot ( r_n, 2*deflection(m,:), '-');
    grid on;
    title  (  '  Chordwise  Mode  Shapes  at  Operational  Rotational  Velocity  for  Uniform
Blade' );

    xlabel ( ' Blade Station (inc)' ) ;

    ylabel ( ' Relative Deflection ' ) ;

    hold on

end % end of for loop count

%

disp('The Natural Frequencies are(cpm): ')

disp(omega_n)

%

disp('The ratio of the Natural Frequency to Rotational Speed is: ')

disp(omega_n/R_V)

omega_rate = omega_n/R_V;

%

% end of function "HINGELESS_NONUNIFORM_EDGEWISE"
```

APPENDIX B. TABLES OF CHARACTERISTIC FUNCTIONS REPRESENTING NORMAL MODES OF VIBRATION OF A BEAM

The tables in [Ref. 18] give the values of the characteristic function and its first three derivatives for each of the first five modes (*n* = 1, 2, 3, 4,5), of three different types of beams. We use only two types of them:

1. Clamped-free Beam
2. Clamped-supported Beam

The functions are tabulated to five decimal places at intervals of the argument corresponding to 1/50 of the beam length, that is, 0.02 *l*.

It may be shown that the characteristic function for a free-supported beam is the same as the second derivative of the characteristic function for a clamped-supported beam. Also the characteristic function for a clamped-free beam is the same as second derivative of the characteristic function for a clamped-clamped beam.[Ref. 18]

A. DATA FOR THE SUPPORTED-FREE BEAM (HINGED BLADE)

Clamped-Supported Beam (Second Derivative)

From [Ref. 18]:

a. First Mode

2.00000, 1.84282, 1.68568, 1.52869, 1.37202,

1.21590, 1.06060, 0.90647, 0.75386, 0.60318,

0.45486, 0.30935, 0.16712, 0.02866,-0.10554,

-0.23500,-0.35923,-0.47775,-0.59009,-0.69582,

-0.79450,-0.88574,-0.96918,-1.04447,-1.11133,

-1.16950,-1.21875,-1.25894,-1.28992,-1.31162,

-1.32402,-1.32714,-1.32106,-1.30588,-1.28180,

-1.24904,-1.20786,-1.15858,-1.10157,-1.03725,

-0.96606,-0.88849,-0.80507,-0.71636,-0.62295,

-0.52547,-0.42455,-0.32086,-0.21507,-0.10789,0

b. Second Mode

2.00000, 1.71729, 1.43502, 1.15424, 0.87658,

0.60415, 0.33937, 0.08494,-0.15633,-0.38158,

-0.58802,-0.77300,-0.93412,-1.06927,-1.17673,

-1.25518,-1.30380,-1.32224,-1.31068,-1.26983,

-1.20092,-1.10569,-0.98634,-0.84553,-0.68631,

-0.51204,-0.32640,-0.13323, 0.06348, 0.25968,

0.45136, 0.63460, 0.80569, 0.96112, 1.09776,

1.21281, 1.30395, 1.36930, 1.40755, 1.41789,

1.40010, 1.35450, 1.28198, 1.18399, 1.06244,

0.91976, 0.75879, 0.58271, 0.39504, 0.19951,0

c. Third Mode

2.00000, 1.59173, 1.18532, 0.78508, 0.39742,

0.03009,-0.30845,-0.60968,-0.86560,-1.06927,

-1.21523,-1.29988,-1.32158,-1.28137,-1.18195,

-1.02863,-0.82967,-0.59110,-0.32637,-0.04596,

0.23807, 0.51362, 0.76897, 0.99330, 1.17711,

1.31263, 1.39411, 1.41807, 1.38344, 1.29160,

1.14631, 0.95356, 0.72134, 0.45927, 0.17821,

-0.11017,-0.39391,-0.66123,-0.90103,-1.10335,

-1.25980,-1.36386,-1.41124,-1.39996,-1.33049,

-1.20573,-1.03085,-0.81313,-0.56162,-0.28677,0

B DATA FOR THE CLAMPED-FREE BEAM (HINGELESS BLADE)

Clamped-Clamped Beam (Second Derivative)

From [Ref. 18]

a. First Mode

2.00000, 1.6861, 1.37287, 1.06189, 0.75558,

0.45702, 0.16974, -0.10243, -0.35563, -0.58594,

-0.78975,-0.96375, -1.10515, -1.21175, -1.28189,

-1.31485, -1.31055, -1.26974, -1.19398, -1.08559,

-0.94753, -0.78359, -0.59802, -0.39555, -0.18130,.

0.03937, 0.26103, 0.47822, 0.68568, 0.87841,

1.05185, 1.20196, 1.32534, 1.41931, 1.48203,

1.51248, 1.51046, 1.47707, 1.41376, 1.32324,

1.20901, 1.07535, 0.92728, 0.77049, 0.61120,

0.45614, 0.31238, 0.18727, 0.08829, 0.02339,0

b. Second Mode

2.0000, 1.80877, 1.61764, 1.42680, 1.23660,

1.04750, 0.86004, 0.67484, 0.49261, 0.31409,

0.14007, -0.02865, -0.19123, -0.34687, -0.49475,

-0.63410, -0.76419, -0.88431, -0.99384, -1.09222,

-1.17895, -1.25365, -1.31600, -1.36578, -1.40289,

-1.42733, -1.43920, -1.43871, -1.42619, -1.40209,

-1.36694, -1.32141, -1.26626, -1.20236, -1.13068,

-1.05227, -0.96827, -0.87992, -0.78852, -0.69544,

-0.60211, -0.51002, -0.42070, -0.33573, -0.25670,

-0.18526, -0.12305, -0.07174, -0.03301, -0.00853,0

c. ***Third Mode***

2.0000, 1.94494, 1.88988, 1.83483, 1.77980,

1.72480, 1.66985, 1.61496, 1.56016, 1.50549,

1.45096, 1.39660, 1.34247, 1.28859, 1.23500,

1.18175, 1.12889, 1.07646, 1.02451, 0.97309,

0.92227, 0.87209, 0.82262, 0.77392, 0.72603,

0.67905, 0.63301, 0.58800, 0.54408, 0.50131,

0.45977, 0.41952, 0.38065, 0.34322, 0.30730,

0.27297, 0.24030, 0.20936, 0.18024, 0.15301,

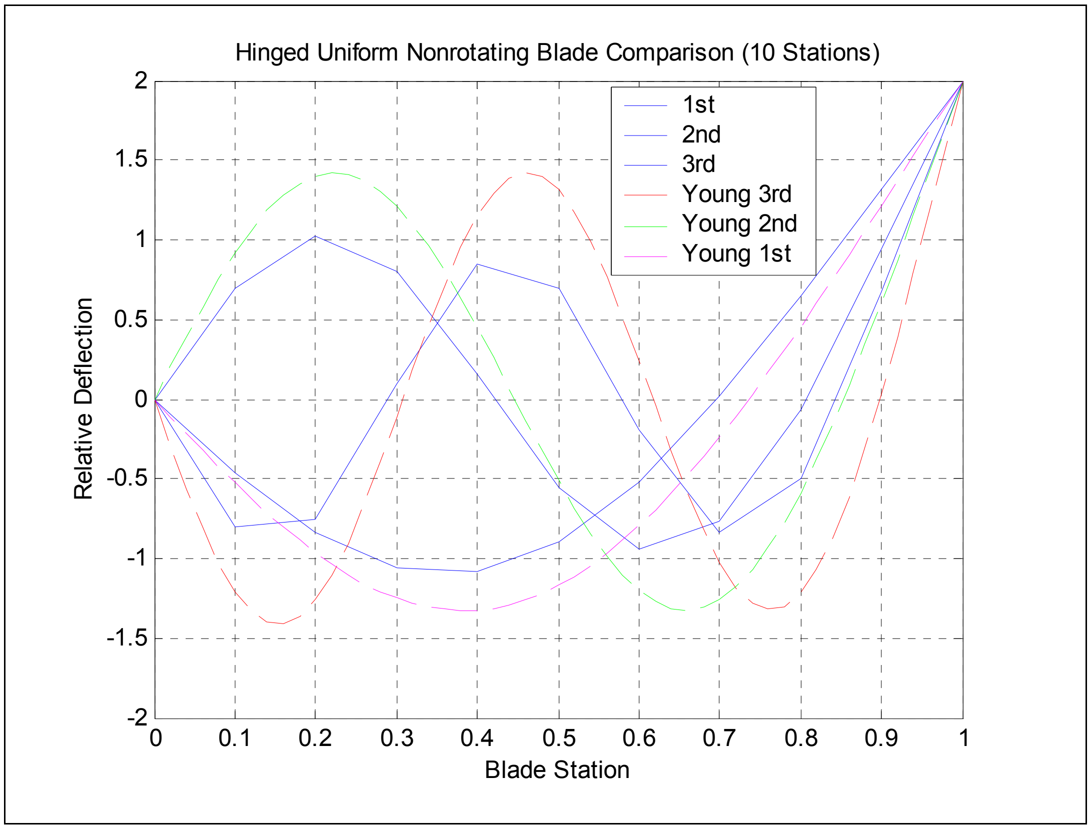
0.12774, 0.10452, 0.08340, 0.06449, 0.04784,

0.03355, 0.02168, 0.01231, 0.00552,0.00139,0

APPENDIX C. VALIDATION OF THE RESULTS OF THE GENERATED MATLAB® CODE

A. VALIDATION OF HINGED NONROTATING UNIFORM BLADE

1. 10 stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0	0.30800000000000	0.48076923076923

According to Young and Felgar the coefficients are:

0	0.30858097836836	0.48043136538462

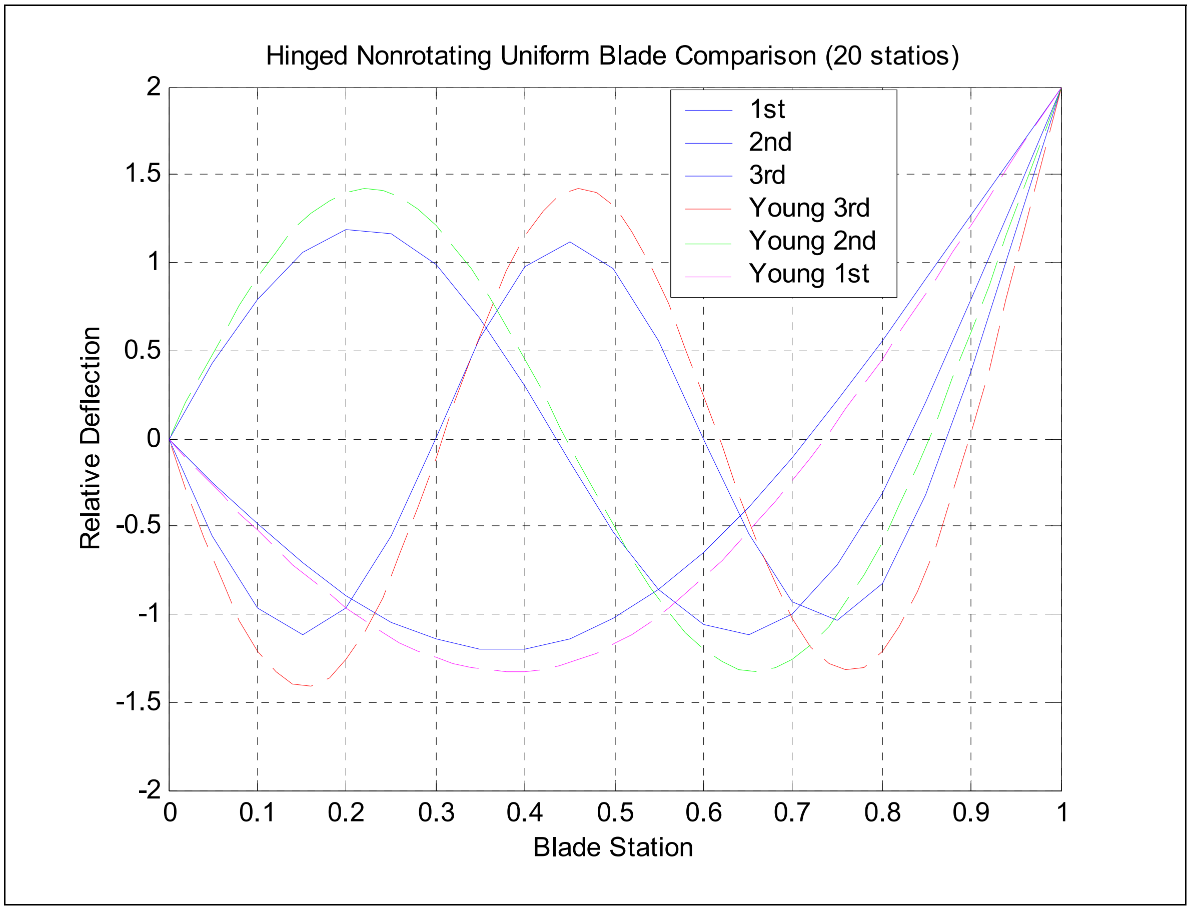
According to Our Program the coefficients are:

0	0.30683909929673	0.47692866436996

The Error ratio of the Natural Frequencies(%)

0	-0.55644803775140	-0.72907417521673

2. 20 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0 0.30800000000000 0.48076923076923

According to Young and Felgar the coefficients are:

0 0.30858097836836 0.48043136538462

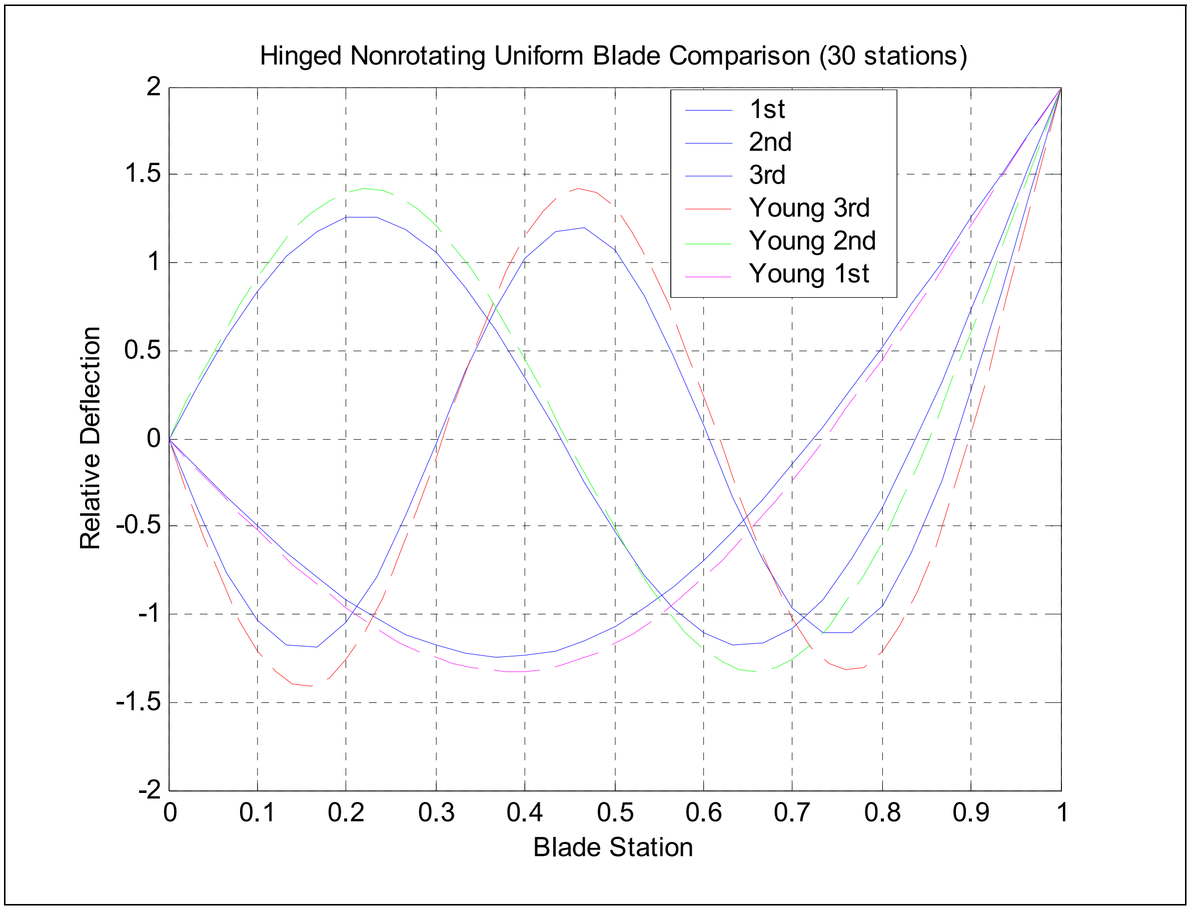
According to Our Program the coefficients are:

0 0.30815824117762 0.47864743924780

The Error ratio of the Natural Frequencies(%)

0 -0.13699392392129 -0.37131758360296

3. 30 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0	0.30800000000000	0.48076923076923

According to Young and Felgar the coefficients are:

0	0.30858097836836	0.48043136538462

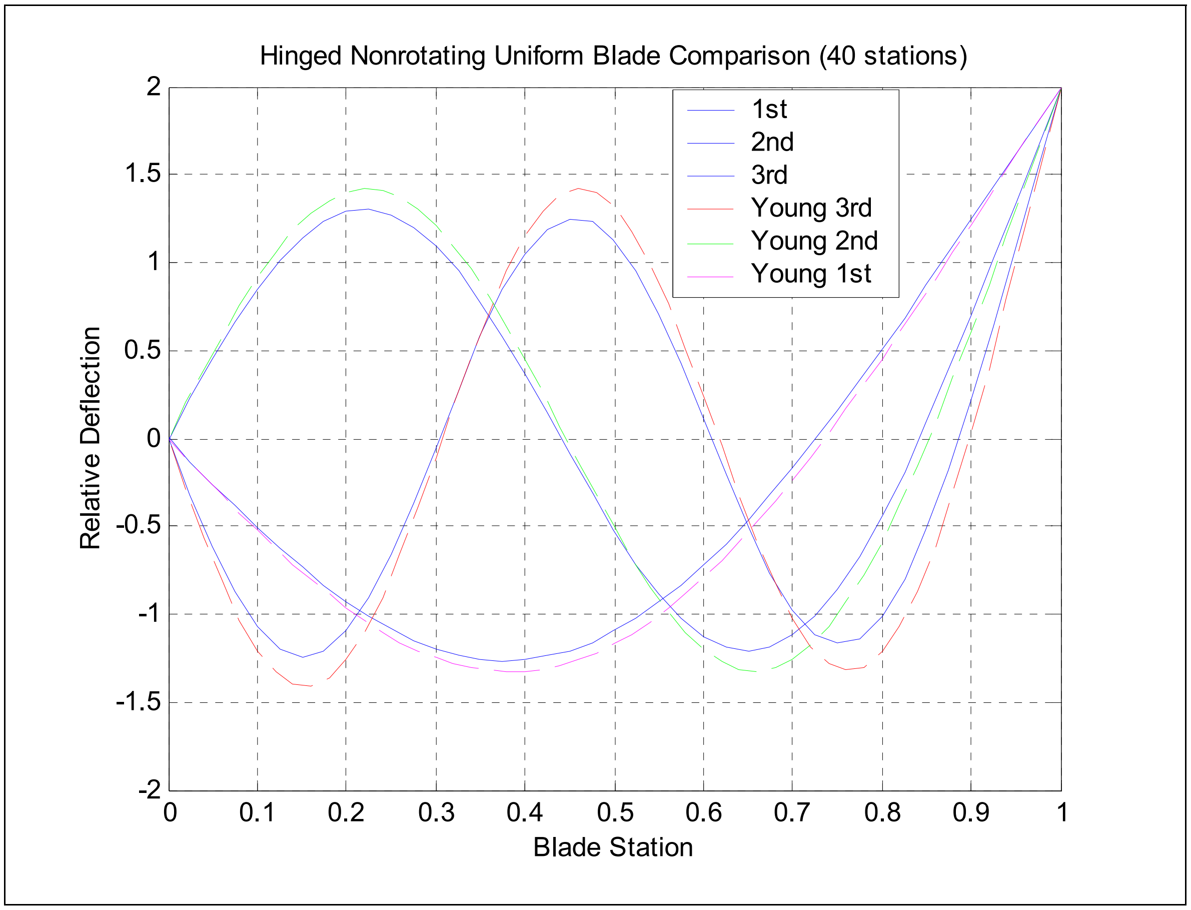
According to Our Program the coefficients are:

0	0.30839579357132	0.47900464866249

The Error Ratio of the Natural Frequencies (%)

0	-0.06001173436615	-0.29696577386870

4. 40 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0	0.30800000000000	0.48076923076923

According to Young and Felgar the coefficients are:

0	0.30858097836836	0.48043136538462

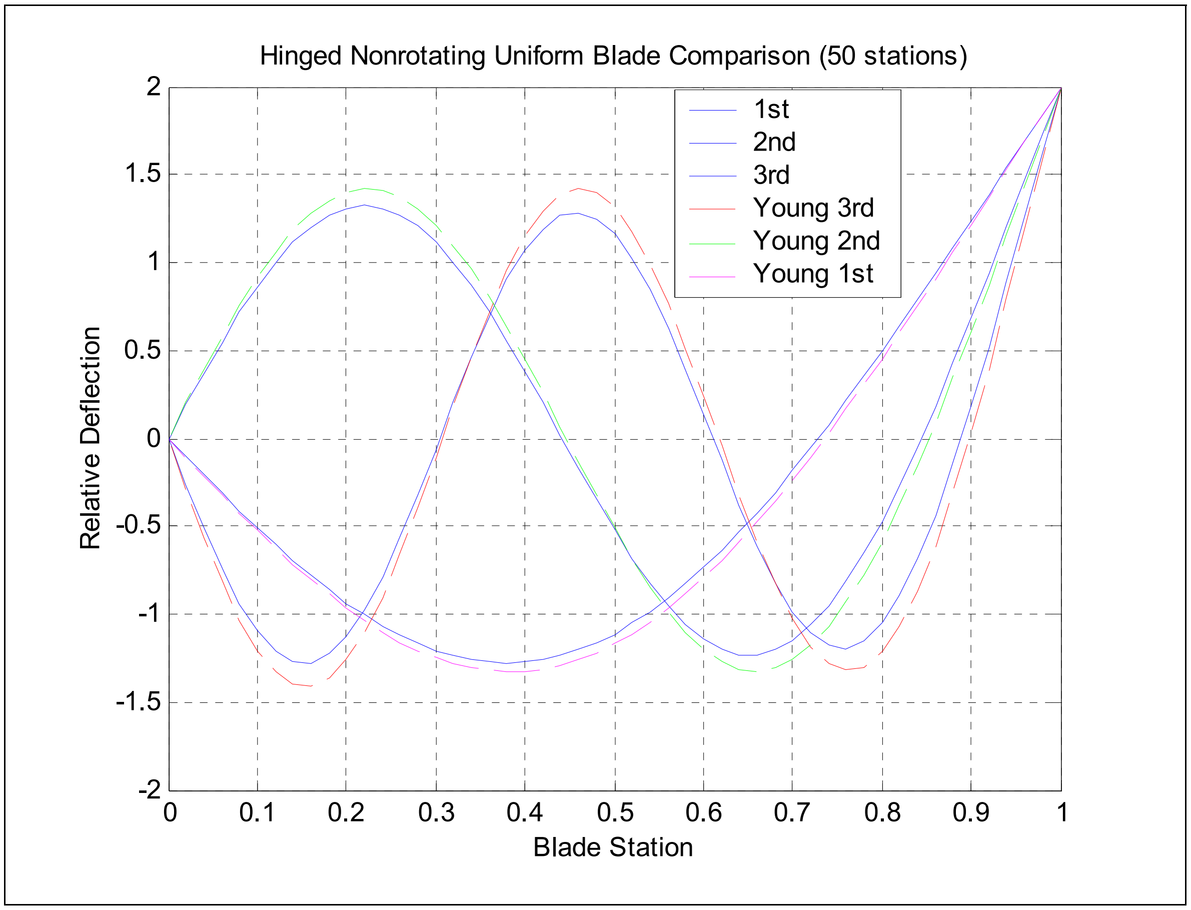
According to Our Program the coefficients are:

0	0.30847776443562	0.47912997196652

The Error Ratio of the Natural Frequencies(%)

0	-0.03344792452286	-0.27088019472901

5. 50 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

```
0               0.30800000000000          0.48076923076923
```

According to Young and Felgar the coefficients are:

```
0               0.30858097836836          0.48043136538462
```

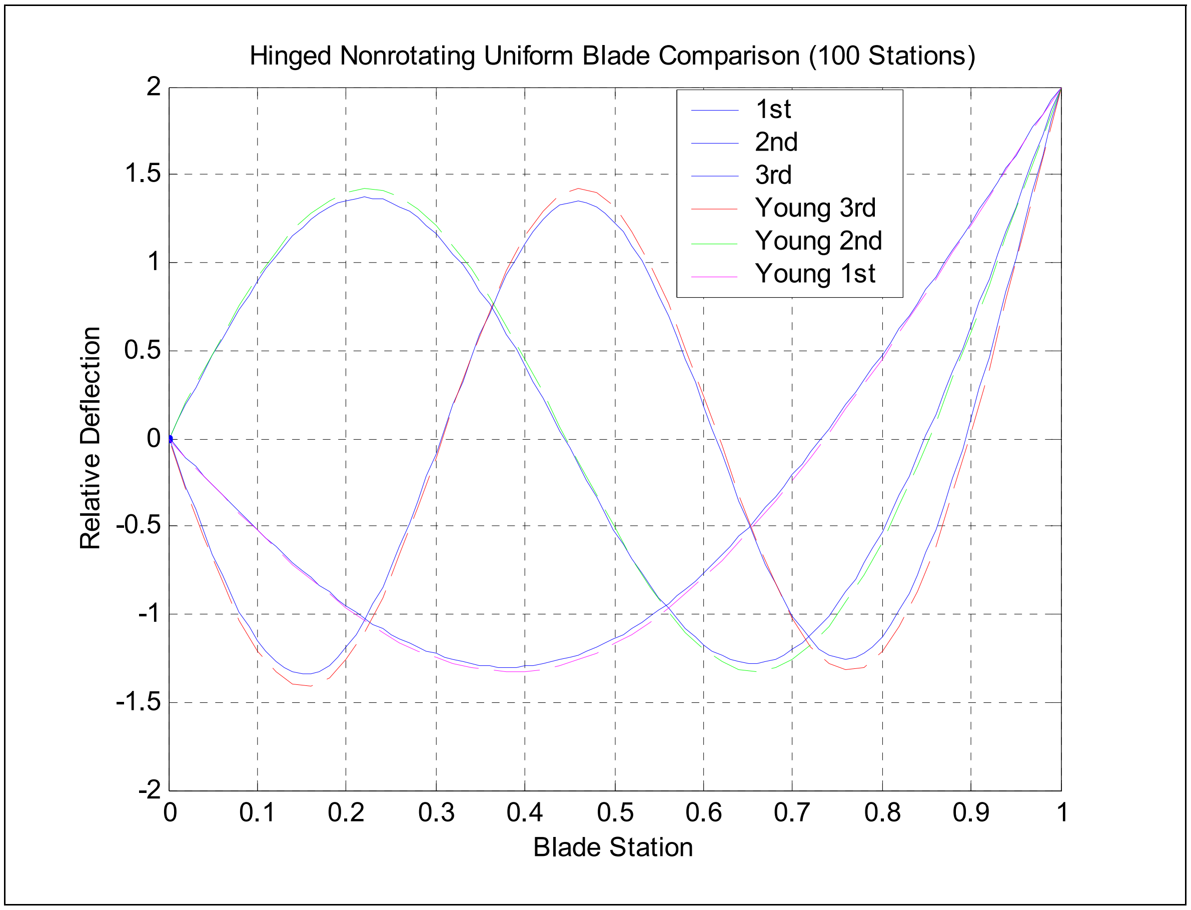
According to Our Program the coefficients are:

```
0               0.30851518704613          0.47918781348356
```

The Error Ratio of the Natural Frequencies(%)

```
0               -0.02132060199573         -0.25884069830882
```

6. 100 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0 0.30800000000000 0.48076923076923

According to Young and Felgar the coefficients are:

0 0.30858097836836 0.48043136538462

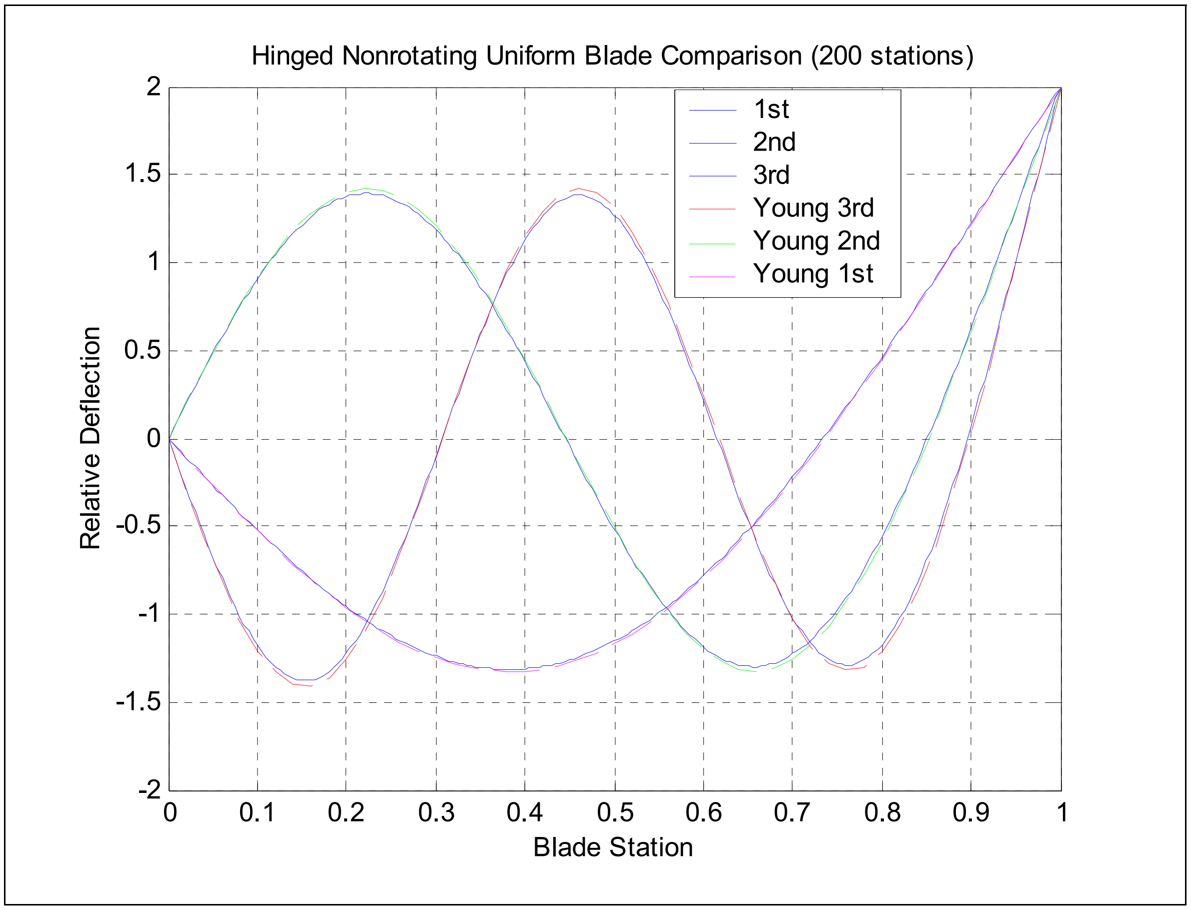
According to Our Program the coefficients are:

0 0.30856485104816 0.47926451848594

The Error Ratio of the Natural Frequencies(%)

0 -0.00522628461558 -0.24287483764511

7. 200 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0 0.30800000000000 0.48076923076923

According to Young and Felgar the coefficients are:

0 0.30858097836836 0.48043136538462

According to Our Program the coefficients are:

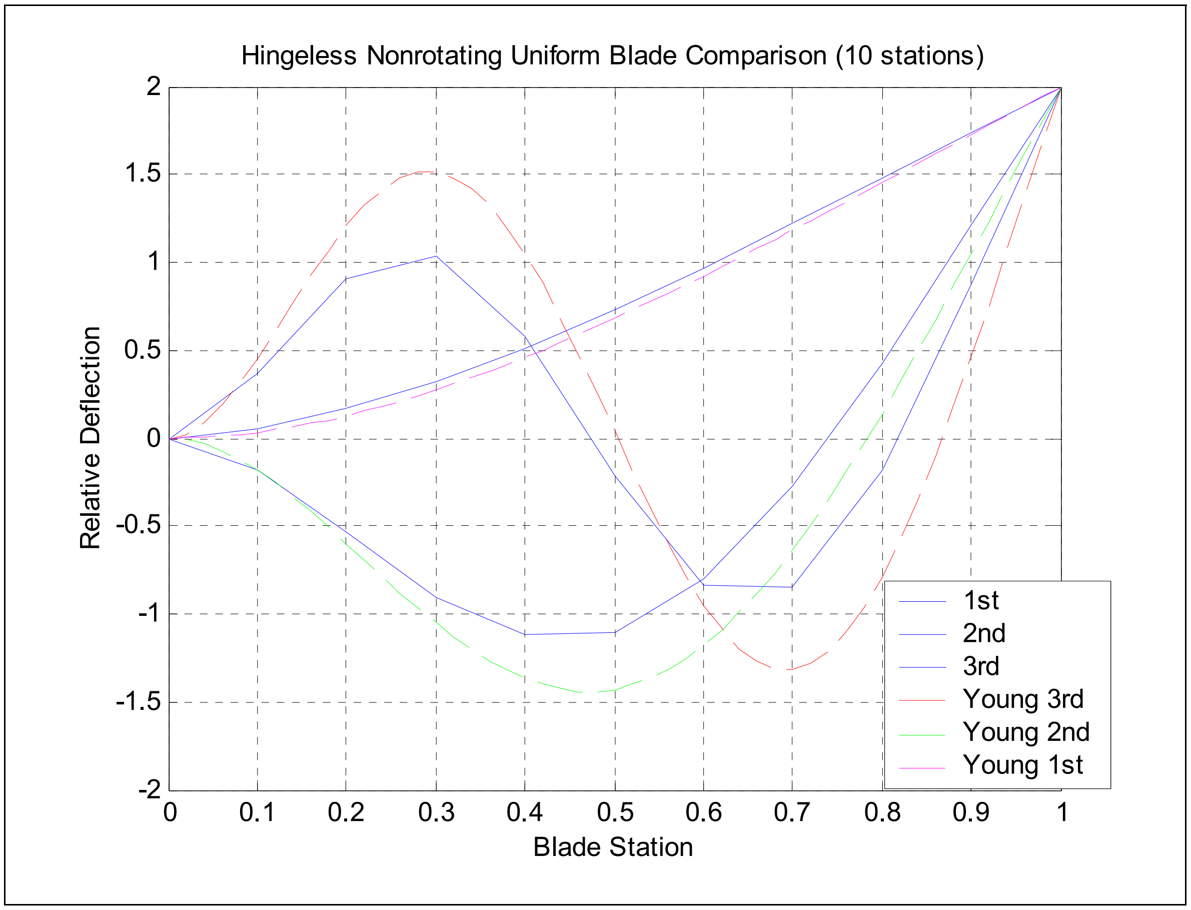
0 0.30857690987193 0.47928353213819

The Error Ratio of the Natural Frequencies(%)

0 -0.00131845340836 -0.23891721671936

B. VALIDATION OF HINGELESS NONROTATING UNIFORM BLADE

1. 10 Staions



Centrifugal Force : 0

According to Hartog the coefficients are:

0.16000000000000 0.35656401944895

According to Young and Felgar the coefficients are:

0.15952332370540 0.35713917325343

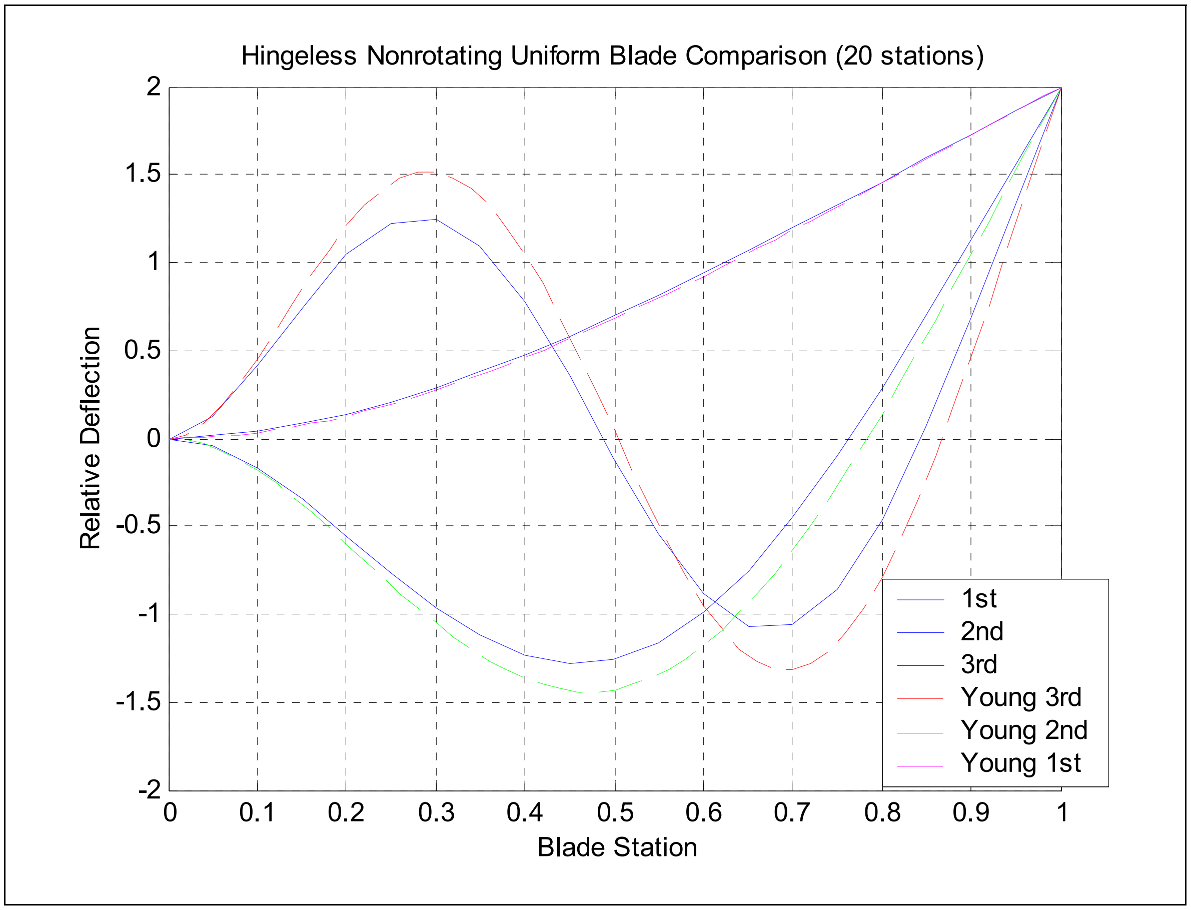
According to Our Program the coefficients are:

0.14893617021277 0.35338345864662

The Error Ratio of the Natural Frequencies(%)

-6.63674329666294 -1.05161093715956

2. 20 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0.16000000000000 0.35656401944895

According to Young and Felgar the coefficients are:

0.15952332370540 0.35713917325343

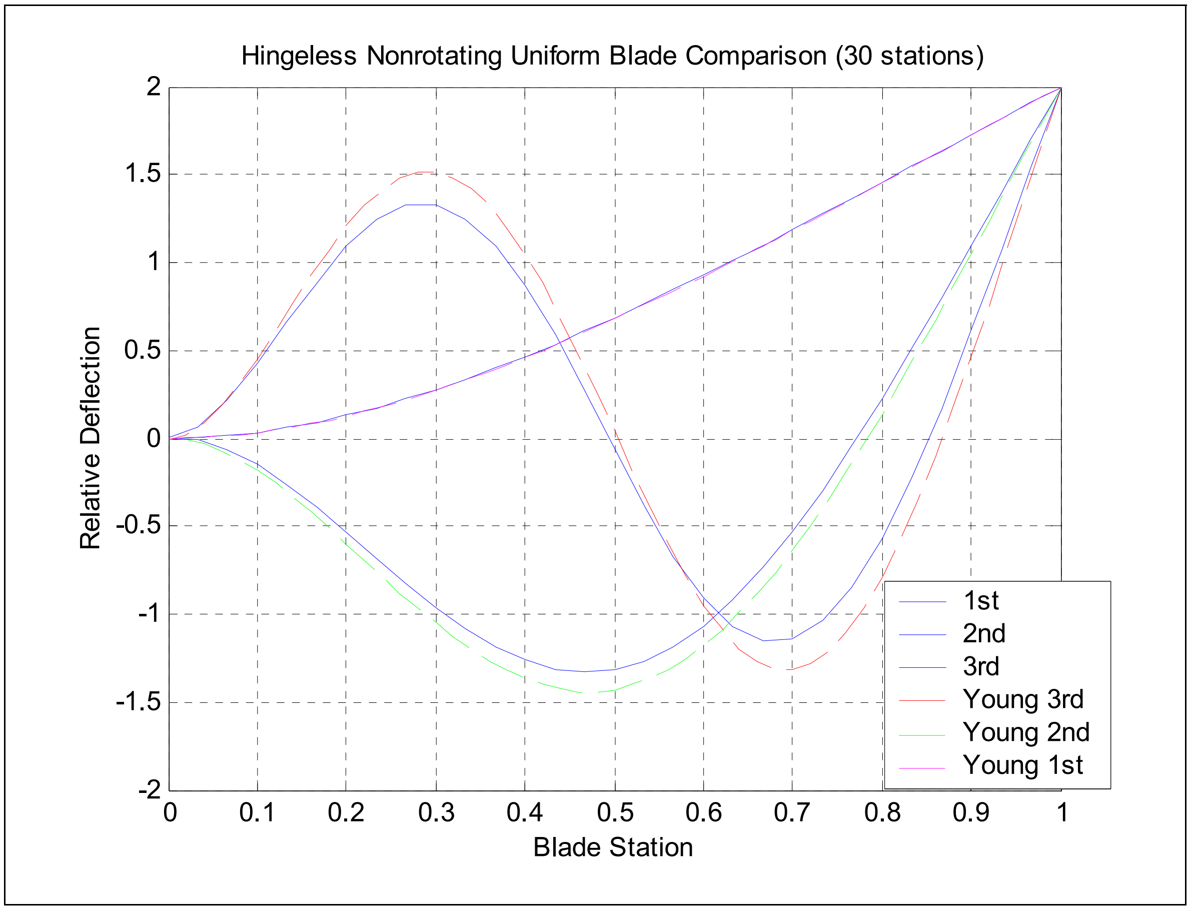
According to Our Program the coefficients are:

0.15555555555556 0.36600000000000

The Error Ratio of the Natural Frequencies(%)

-2.48726522095906 2.45007975167658

3. 30 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0.16000000000000   0.35656401944895

According to Young and Felgar the coefficients are:

0.15952332370540   0.35713917325343

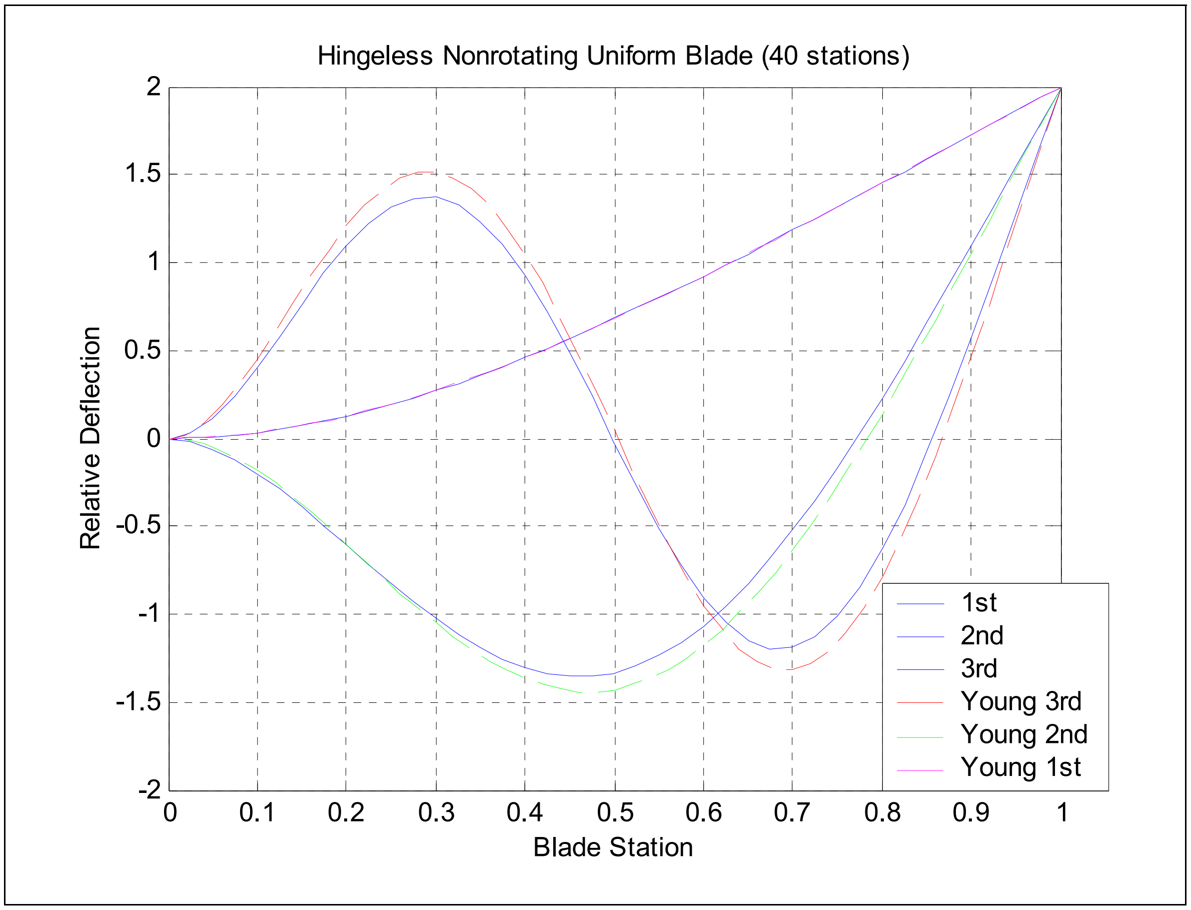
According to Our Program the coefficients are:

0.15555555555556   0.36585365853659

The Error Ratio of the Natural Frequencies(%)

-2.48726522095906   2.44008104845181

4. 40 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0.16000000000000 0.35656401944895

According to Young and Felgar the coefficients are:

0.15952332370540 0.35713917325343

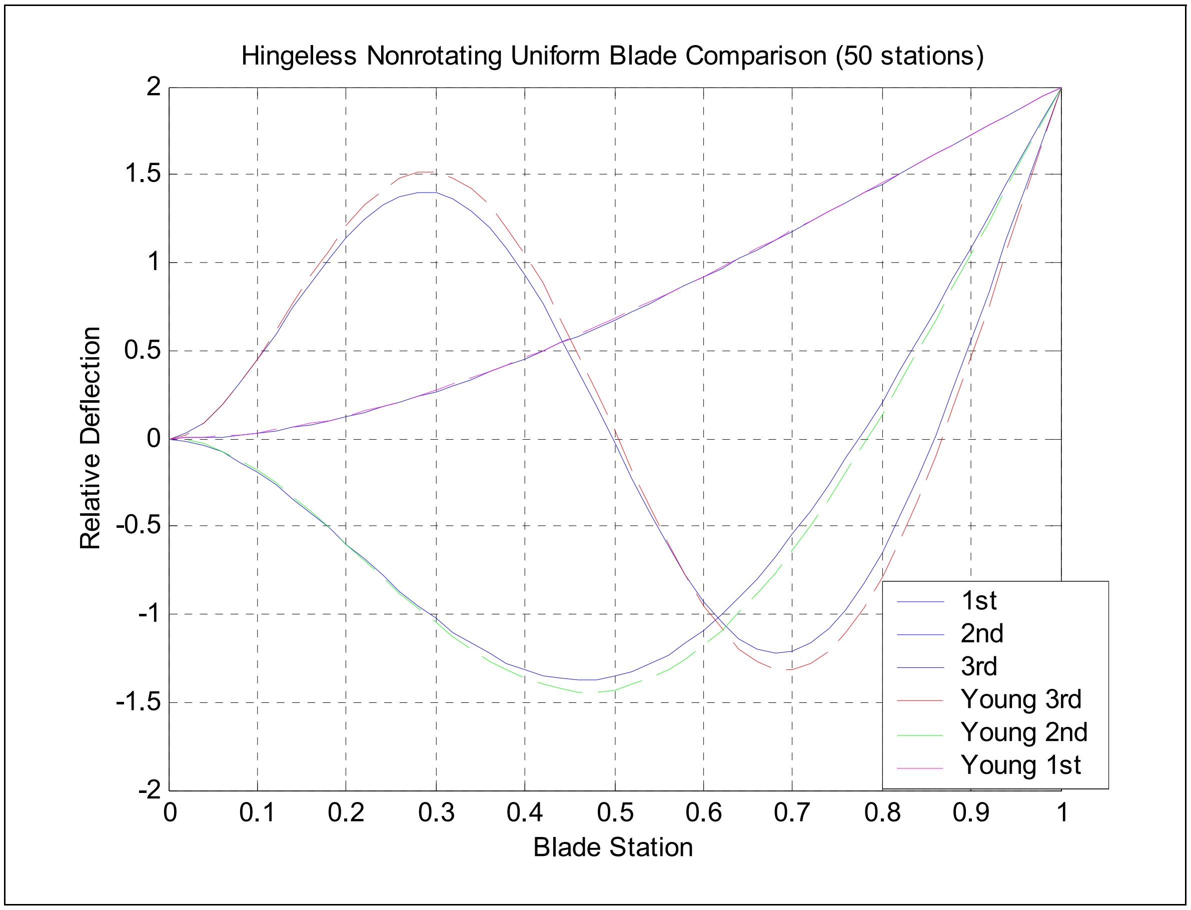
According to Our Program the coefficients are:

0.16279069767442 0.34959349593496

The Error Ratio of the Natural Frequencies(%)

2.04821081527540 -2.11281144259047

5. 50 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0.16000000000000 0.35656401944895

According to Young and Felgar the coefficients are:

0.15952332370540 0.35713917325343

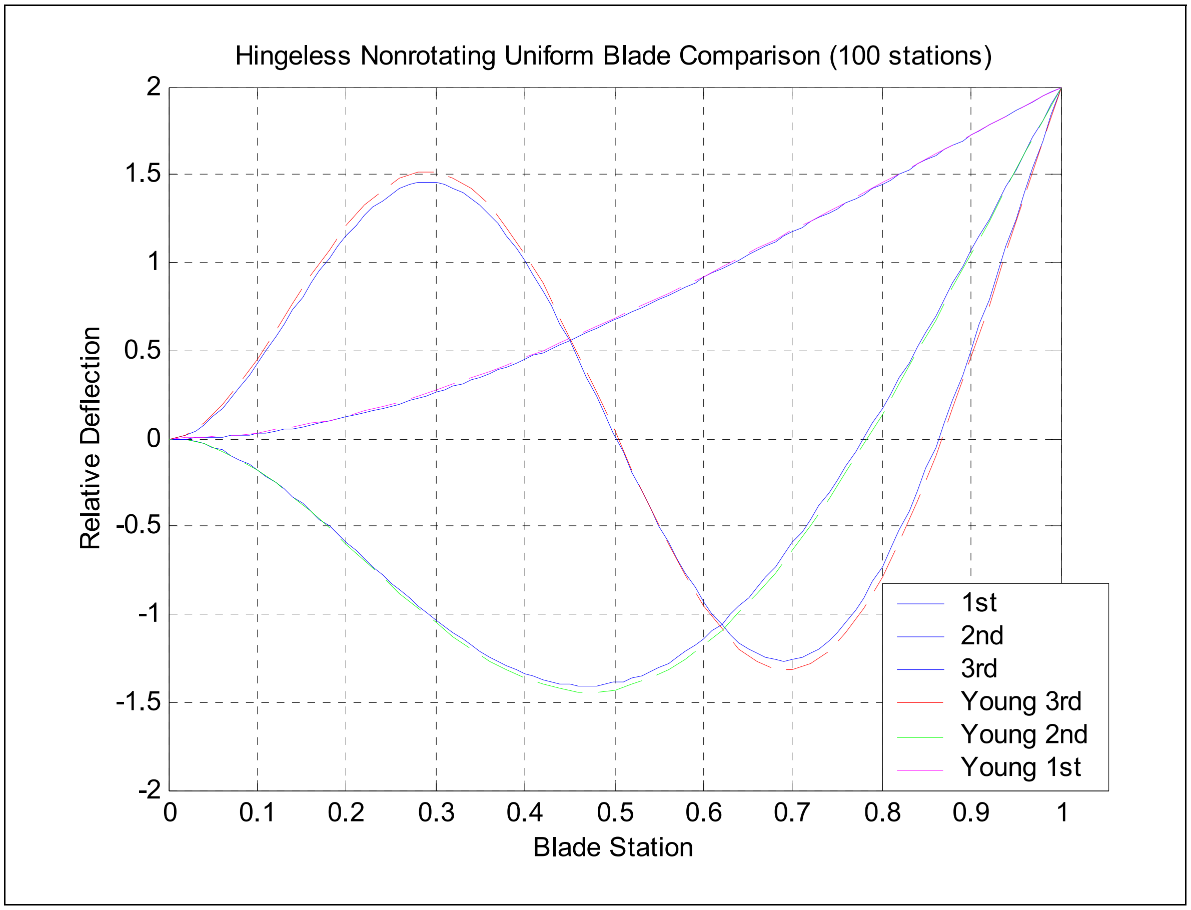
According to Our Program the coefficients are:

0.16279069767442 0.35537190082645

The Error Ratio of the Natural Frequencies(%)

2.04821081527540 -0.49484138379032

6. 100 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0.16000000000000 0.35656401944895

According to Young and Felgar the coefficients are:

0.15952332370540 0.35713917325343

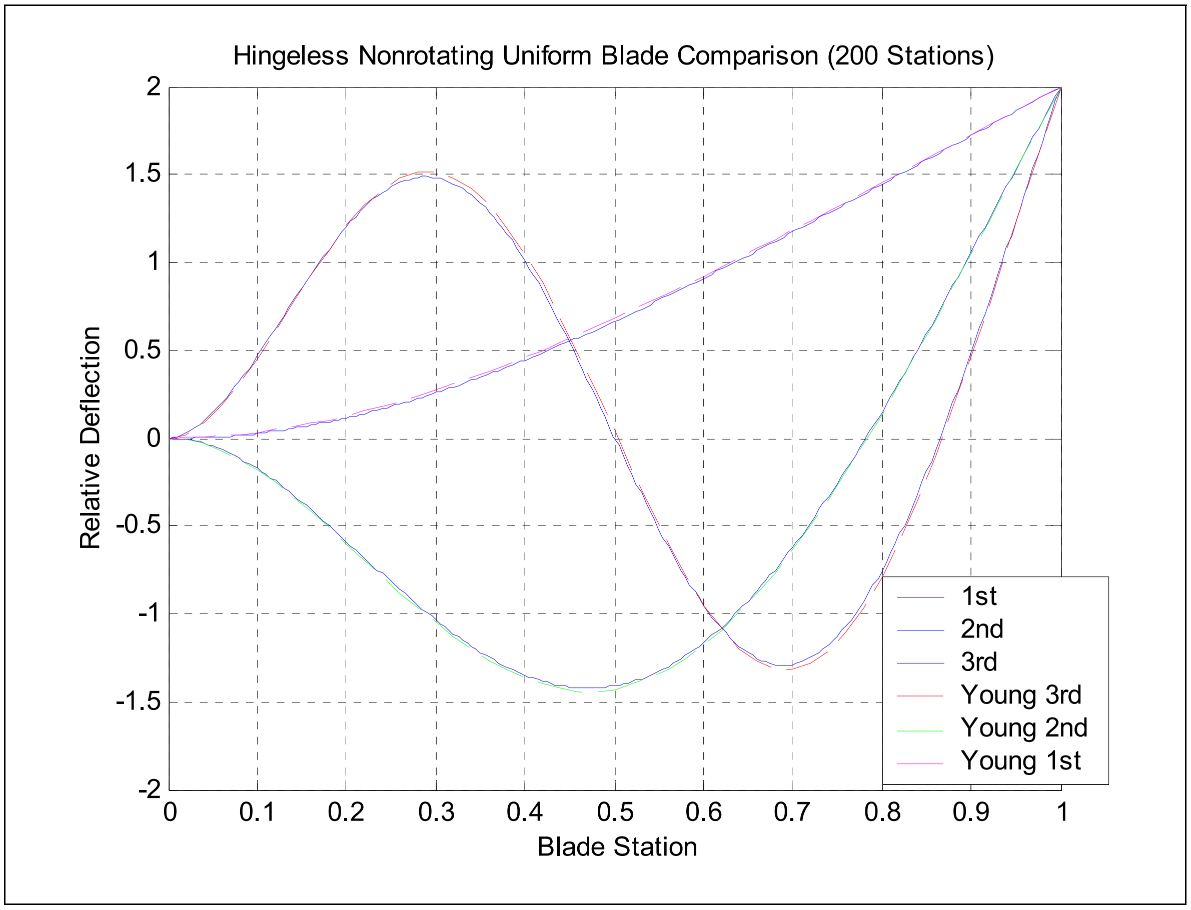
According to Our Program the coefficients are:

0.16279069767442 0.35537190082645

The Error Ratio of the Natural Frequencies(%)

2.04821081527540 -0.49484138379032

7. 200 Stations



Centrifugal Force : 0

According to Hartog the coefficients are:

0.16000000000000 0.35656401944895

According to Young and Felgar the coefficients are:

0.15952332370540 0.35713917325343

According to Our Program the coefficients are:

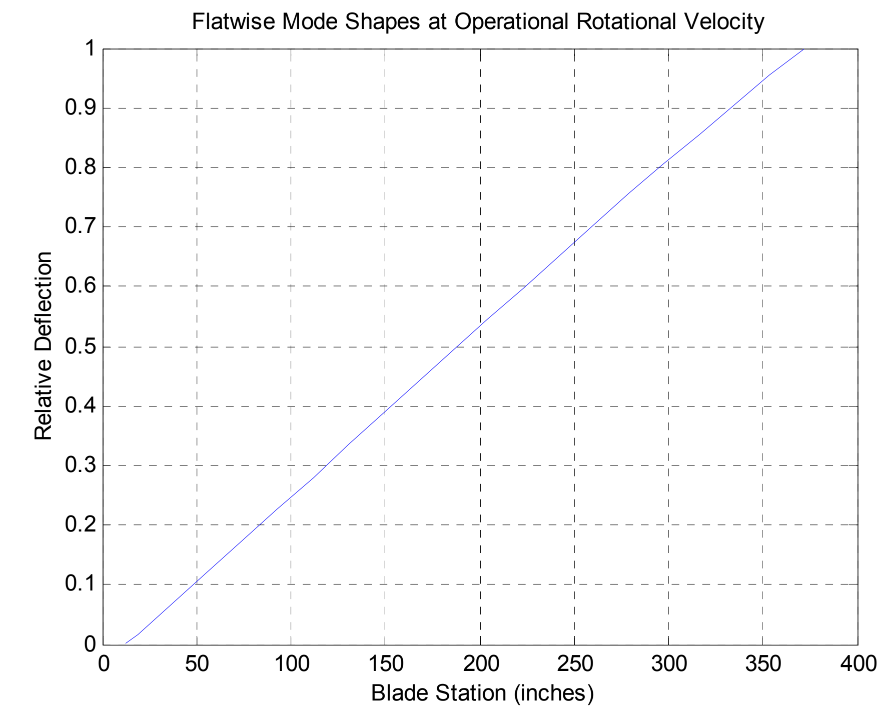
0.16279069767442 0.36134453781513

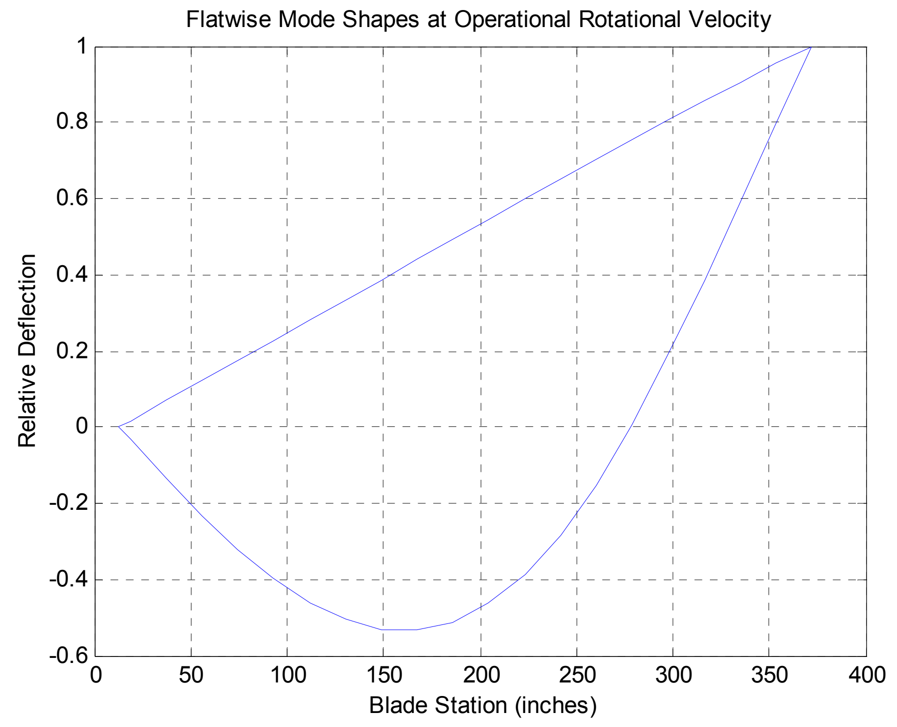
The Error Ratio of the Natural Frequencies(%)

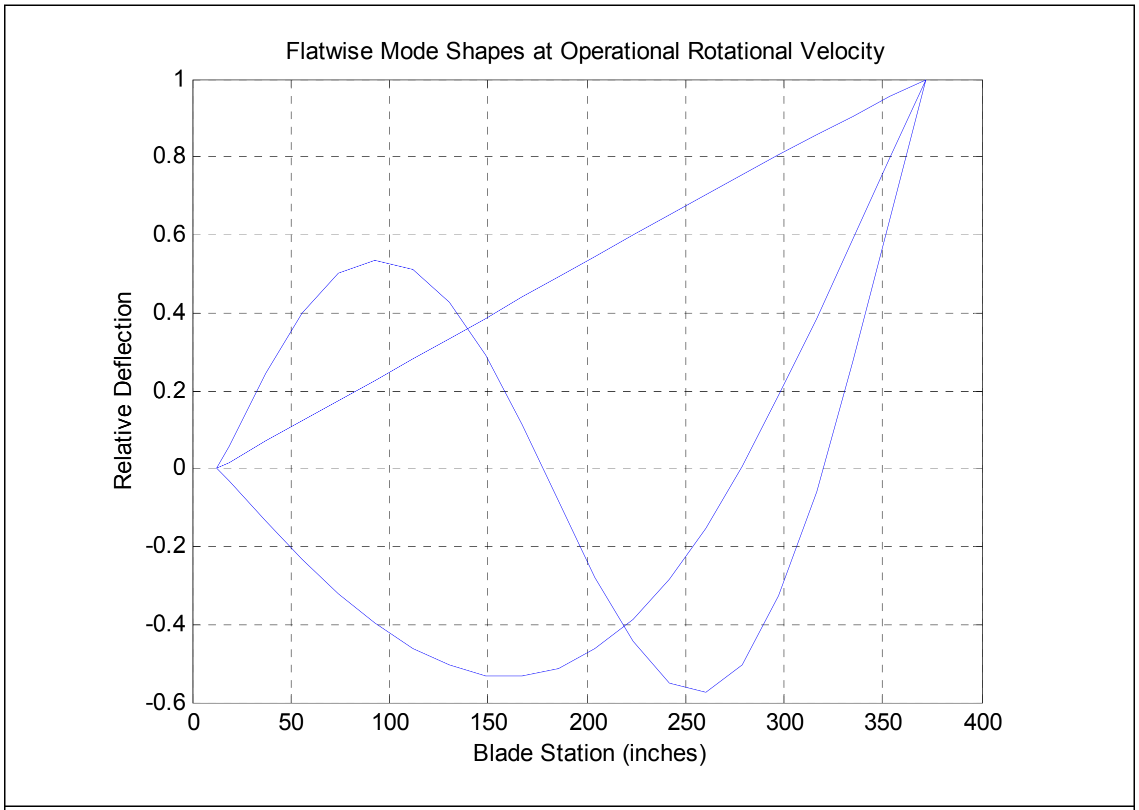
2.04821081527540 1.17751422320480

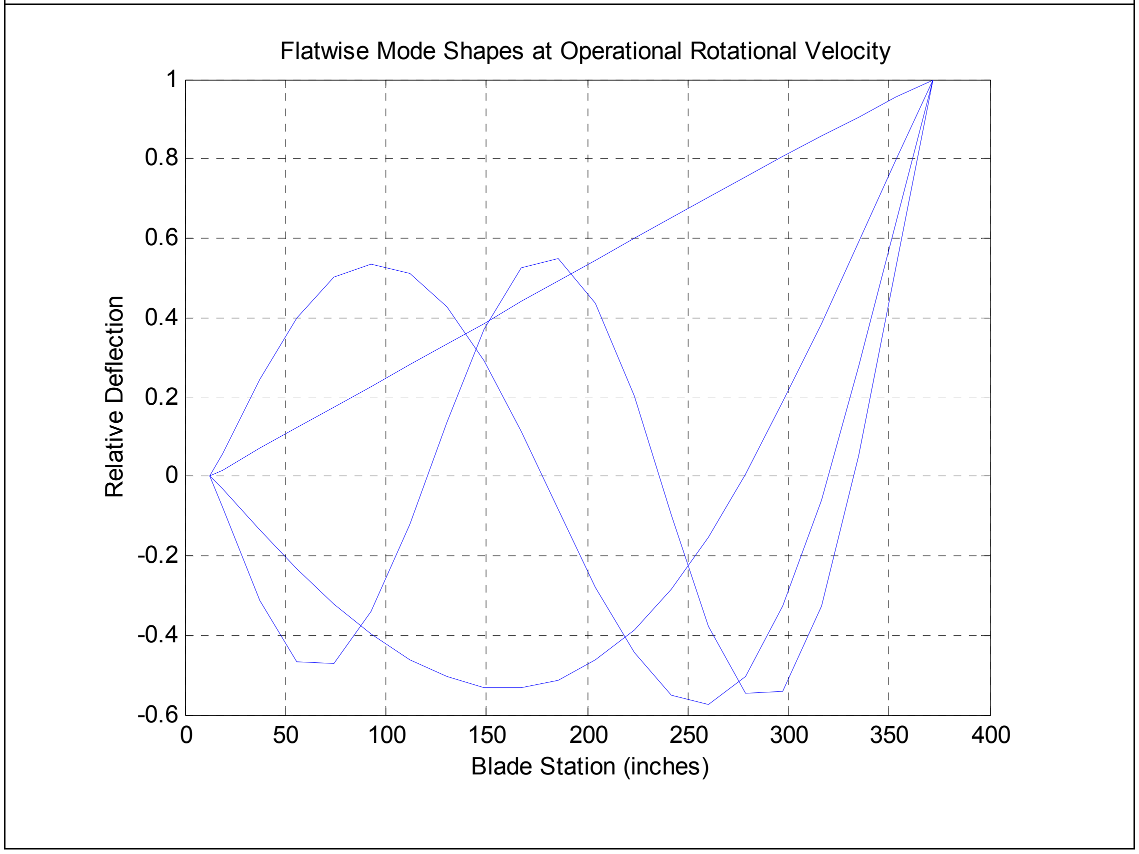
C. RESULTS FOR H-3 HELICOPTER FOR ANALYSIS OF ROTATING NONUNIFORM HINGED BLADES

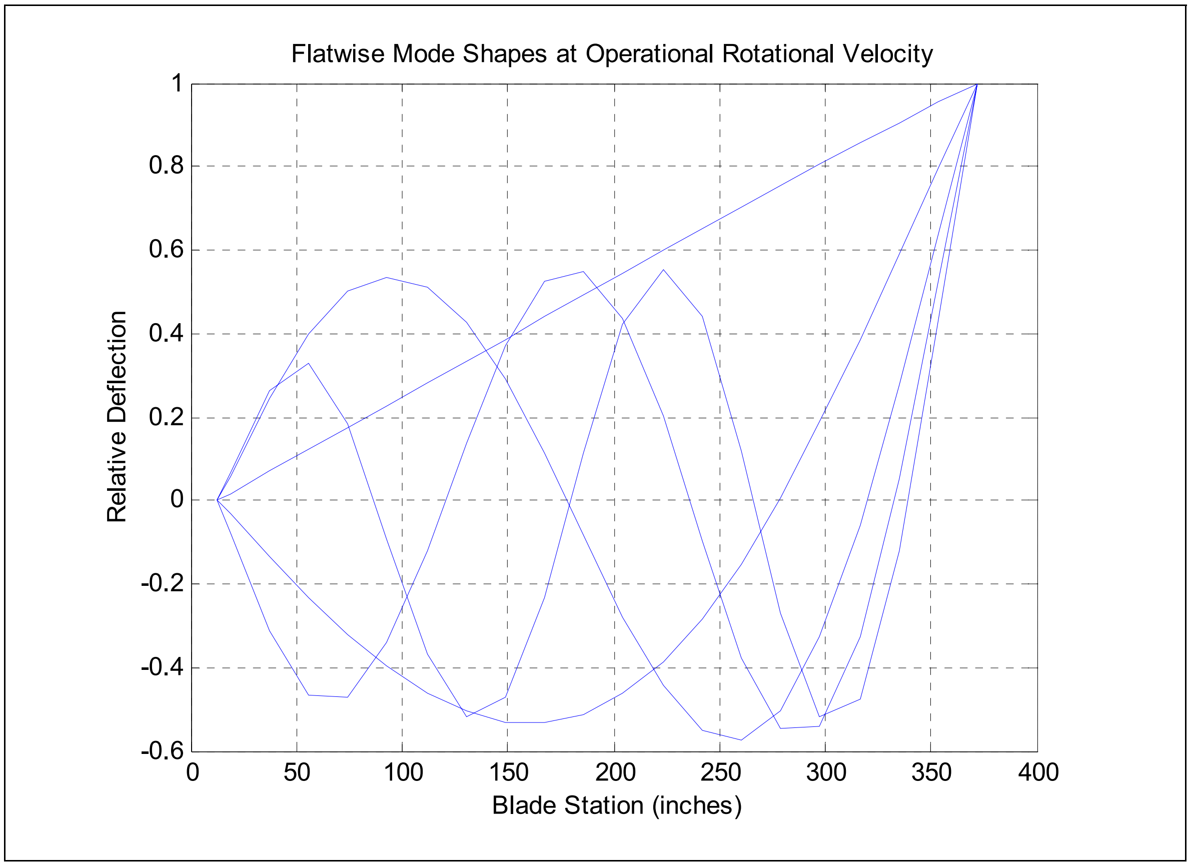
1. Flatwise Mode Shapes






Flatwise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inches)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400


Flatwise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inches)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400

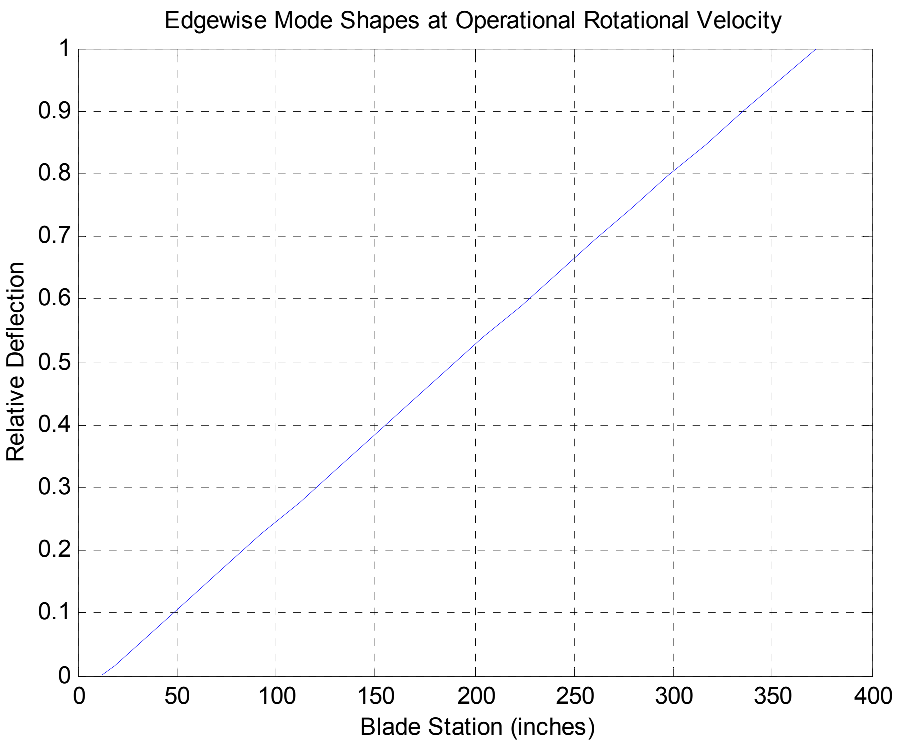


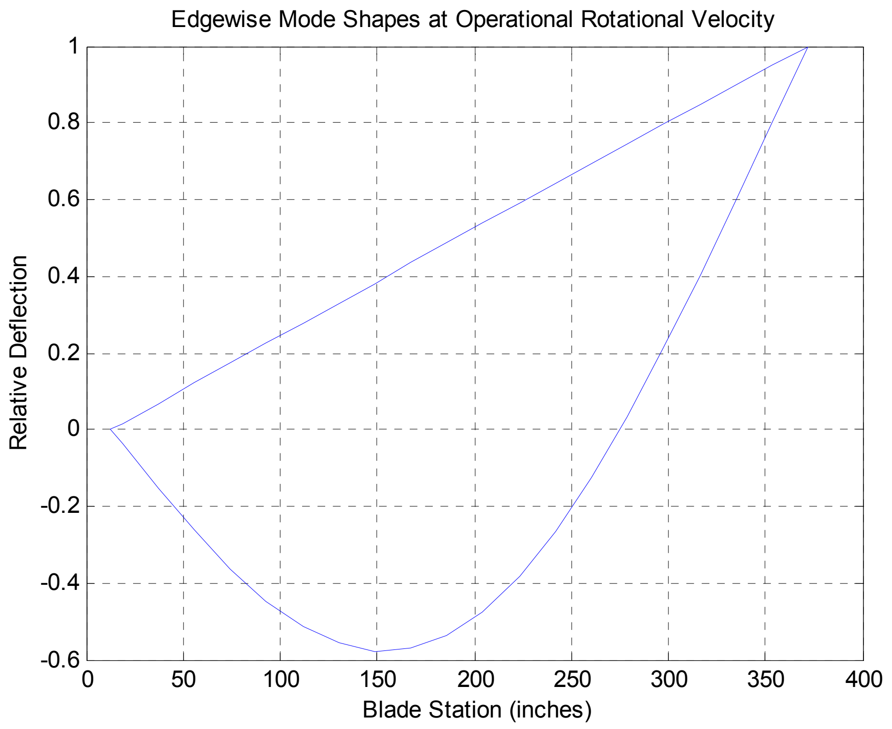
<u>Centrifugal Force is :</u>
4.515809216498805e+004

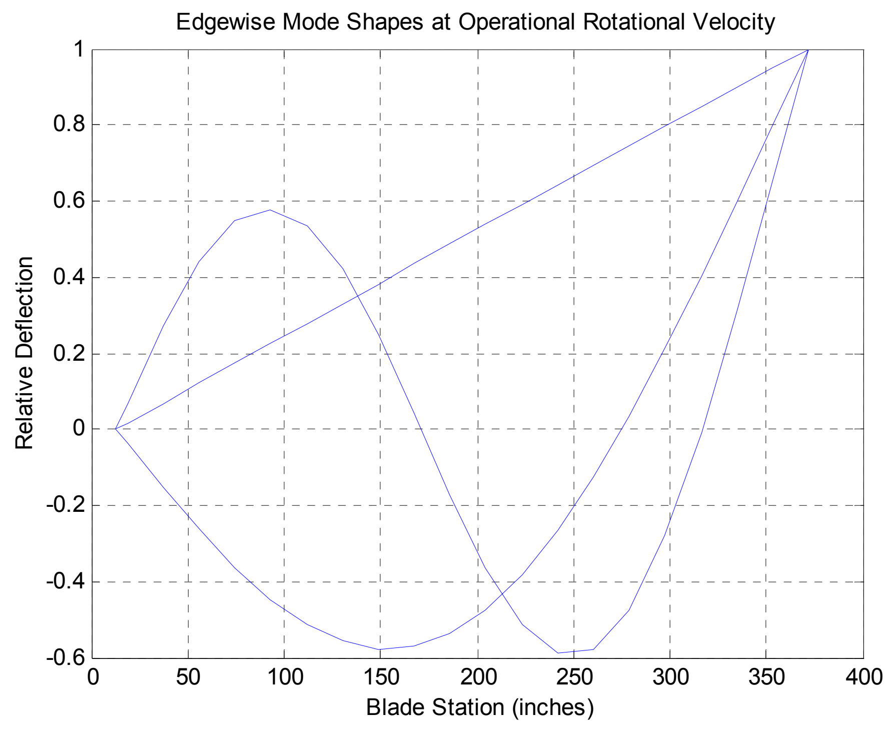
<u>The Natural Frequencies are(cpm):</u>
212.94221992493
554.01301746368
1006.88151531219
1575.99021453857
2341.91161987331

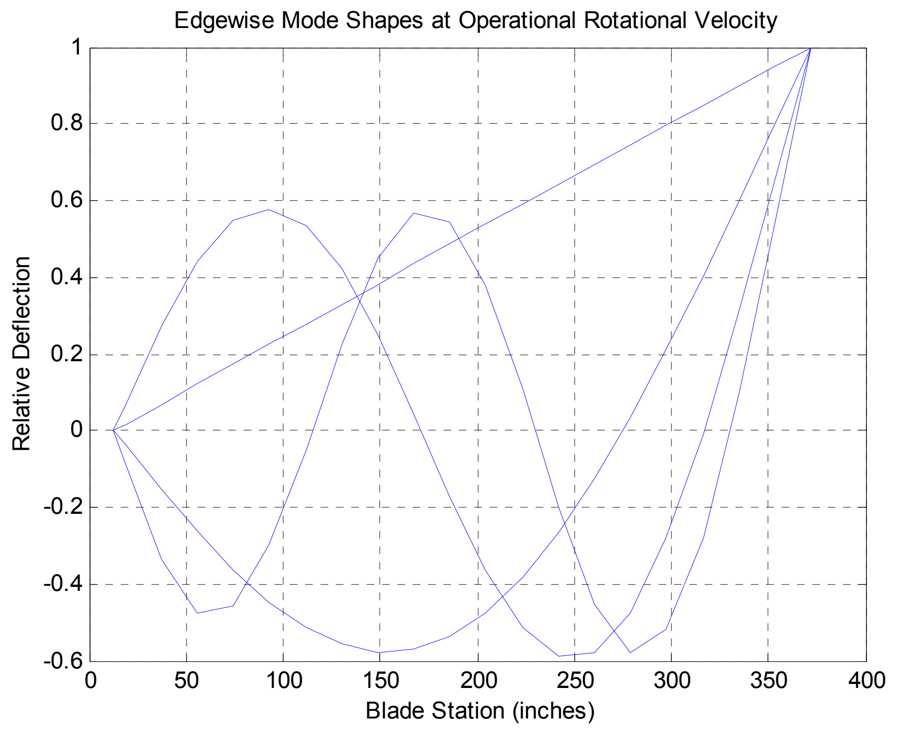
<u>The ratio of the Natural Frequency to Rotational Speed is:</u>
1.04897645283215
2.72912816484574
4.96000746459209
7.76349859378608
11.53651044272568

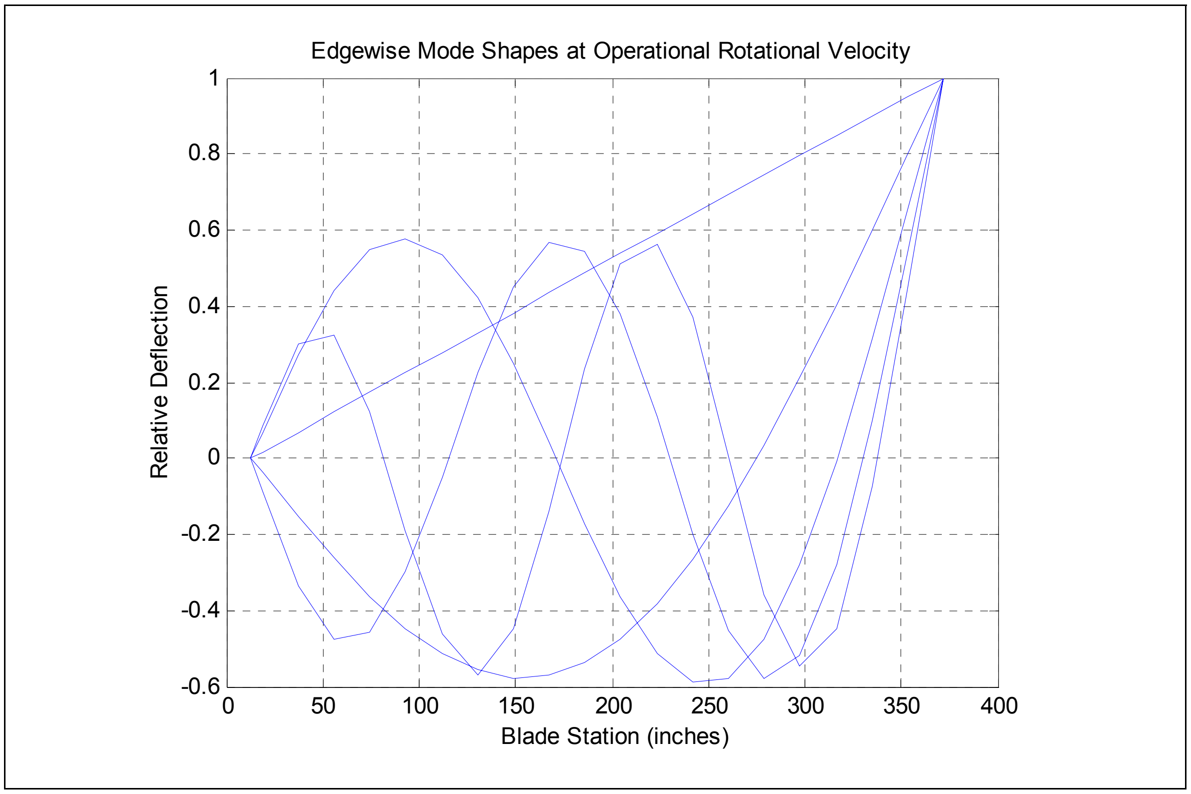
2. Edgewise Mode Shapes






Edgewise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inches)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400


Edgewise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inches)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400



Centrifugal Force :
4.515809216498805e+004

The Natural Frequencies are(cpm):

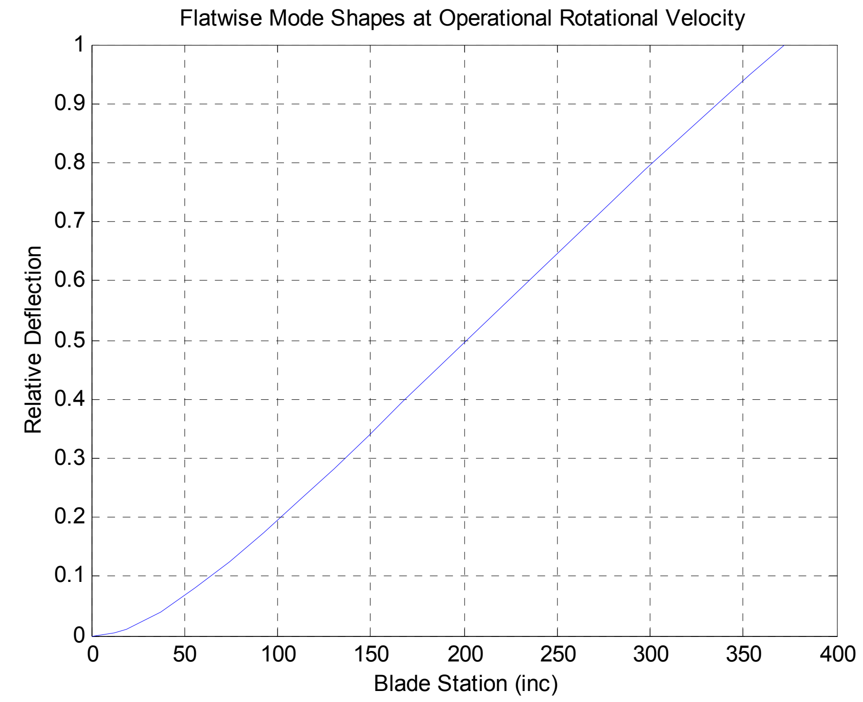
64.46092529297
713.63865966797
1828.18133850098
3443.59081058502
5693.24688033461

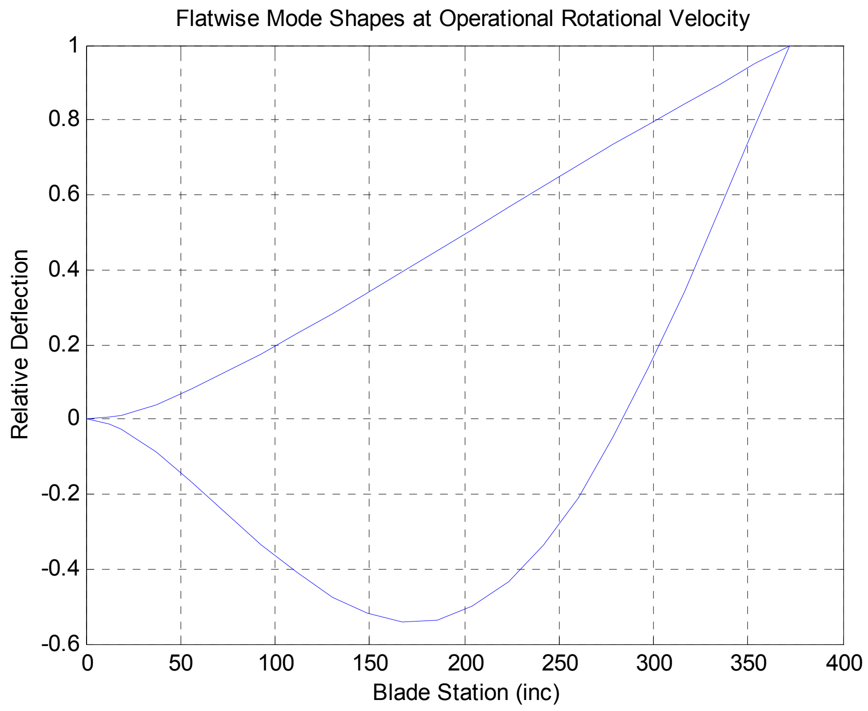
The ratio of the Natural Frequency to Rotational Speed is:

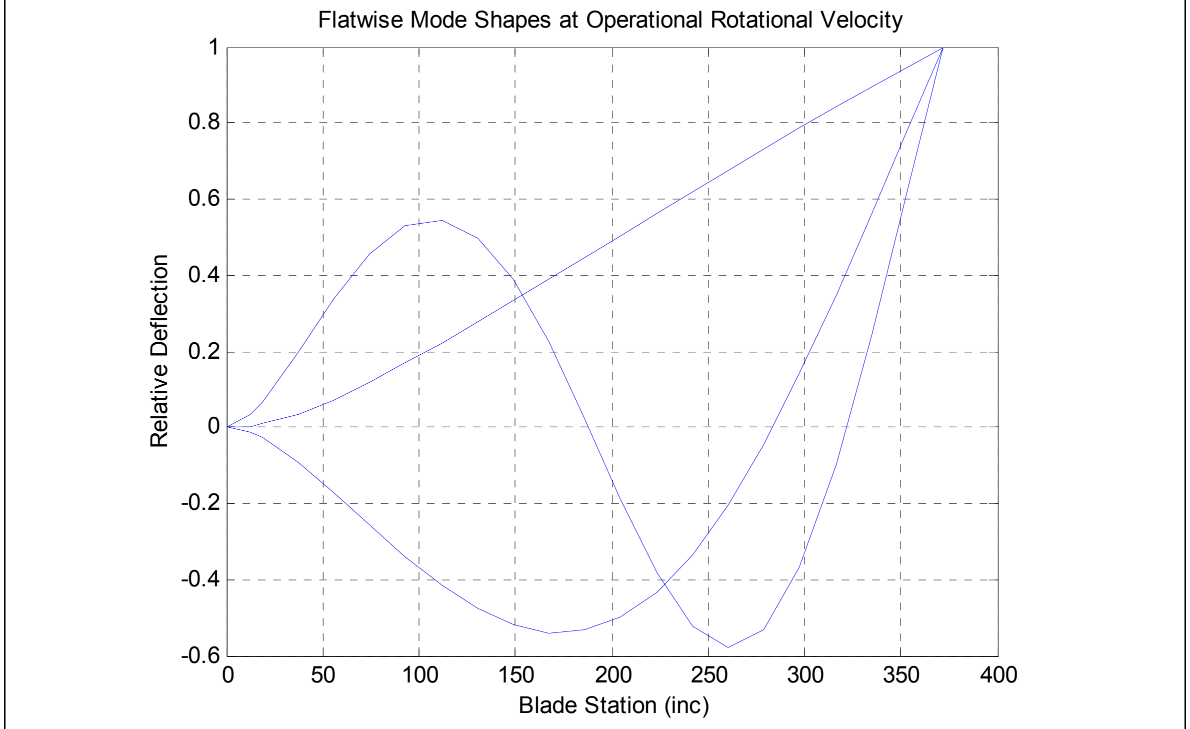
0.31754150390625
3.51546137767472
9.00581940148264
16.96350152997548
28.04555113465327

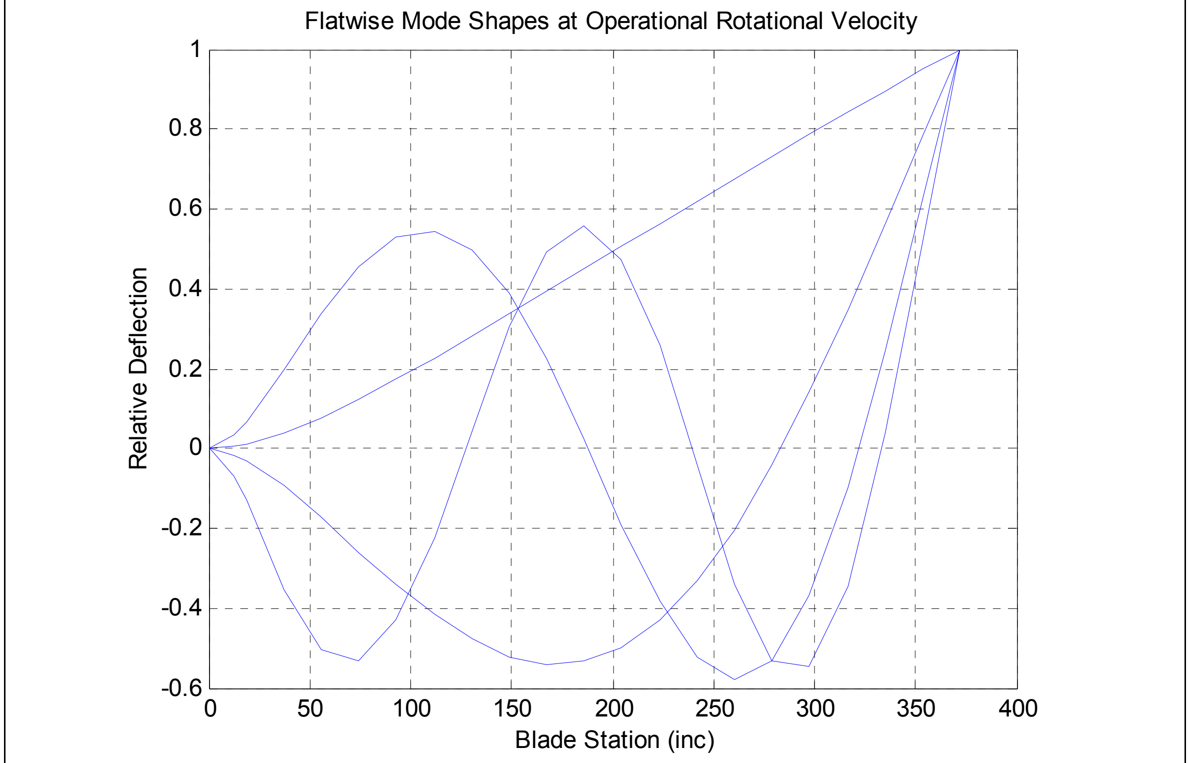
D. RESULTS FOR H-3R HELICOPTER FOR ANALYSIS OF ROTATING NONUNIFORM HINGELESS BLADES

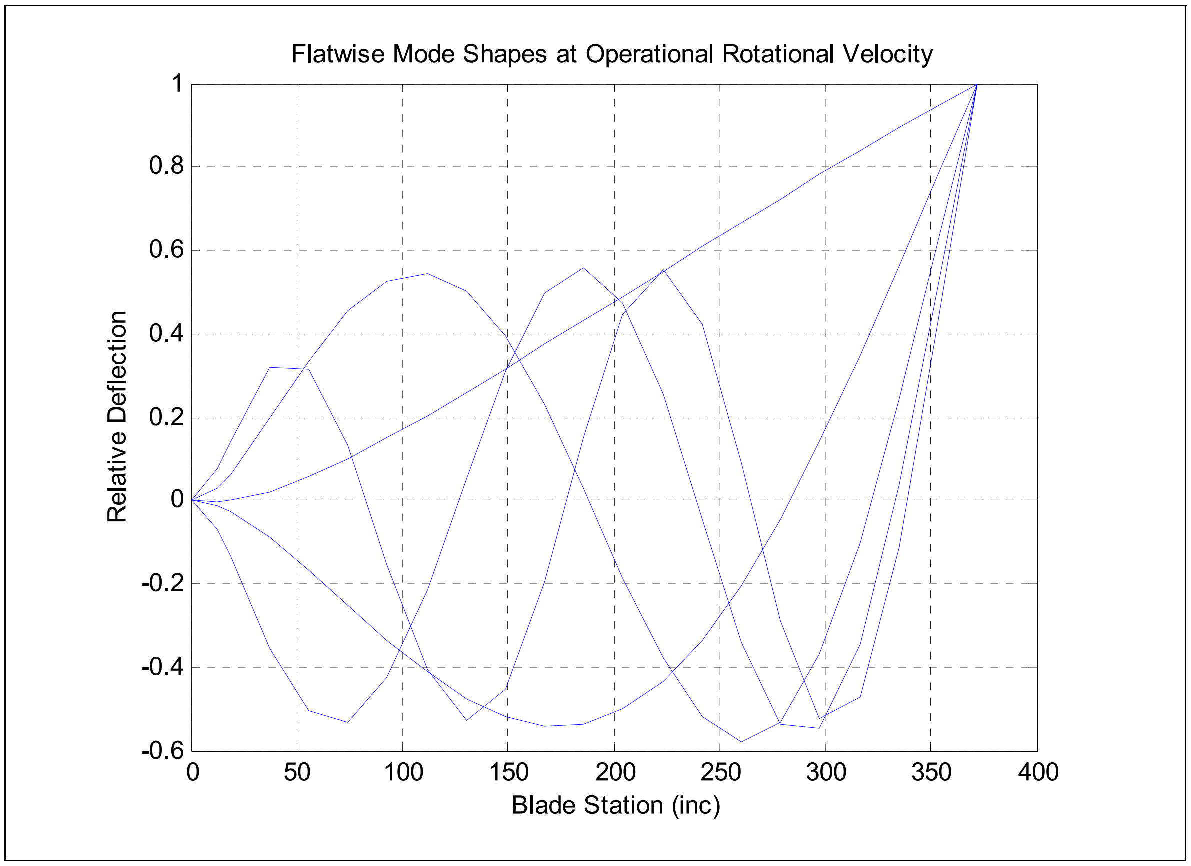
1. Flatwise Mode Shapes






Flatwise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inc)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400


Flatwise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inc)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400



<u>Centrifugal Force :</u>
4.515809216498805e+004

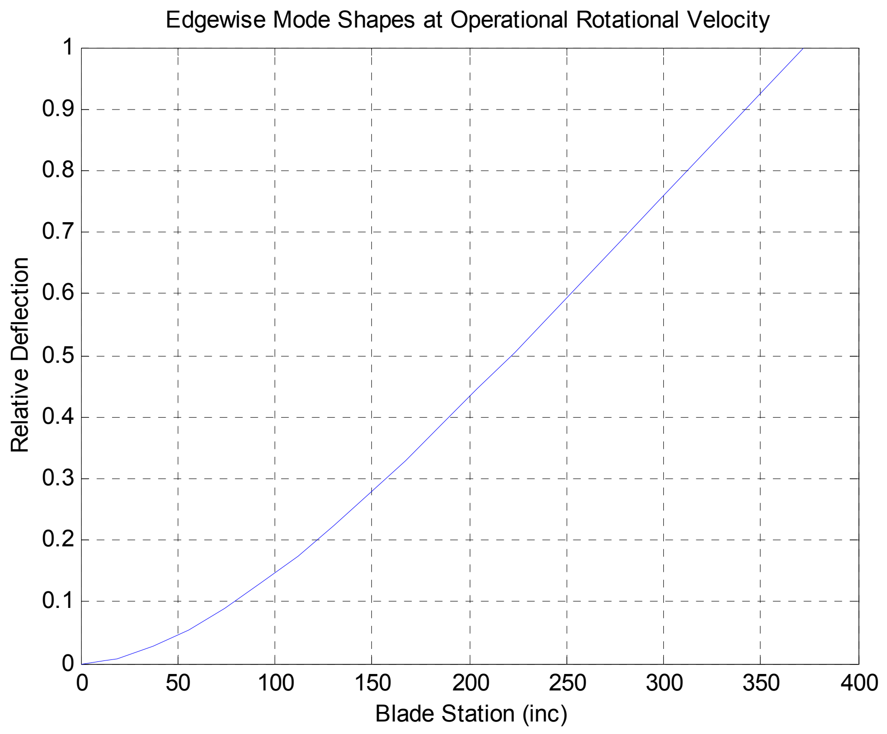
<u>The Natural Frequencies are(cpm):</u>

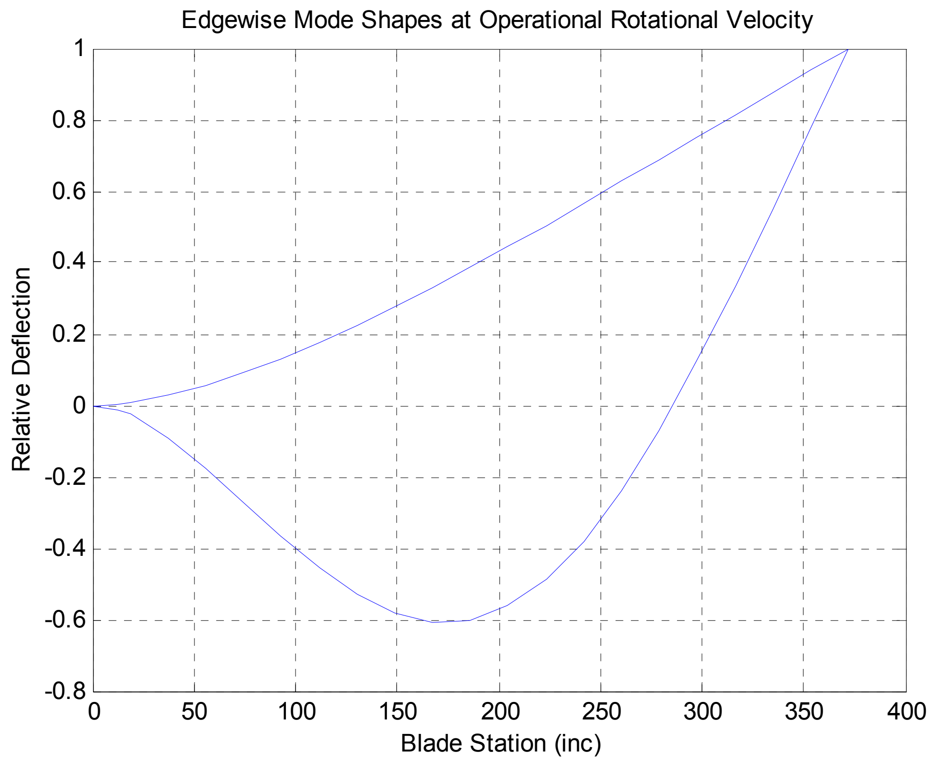
228.37500000000
583.62500000000
1070.82500000000
1629.07500000000
2298.97500000000

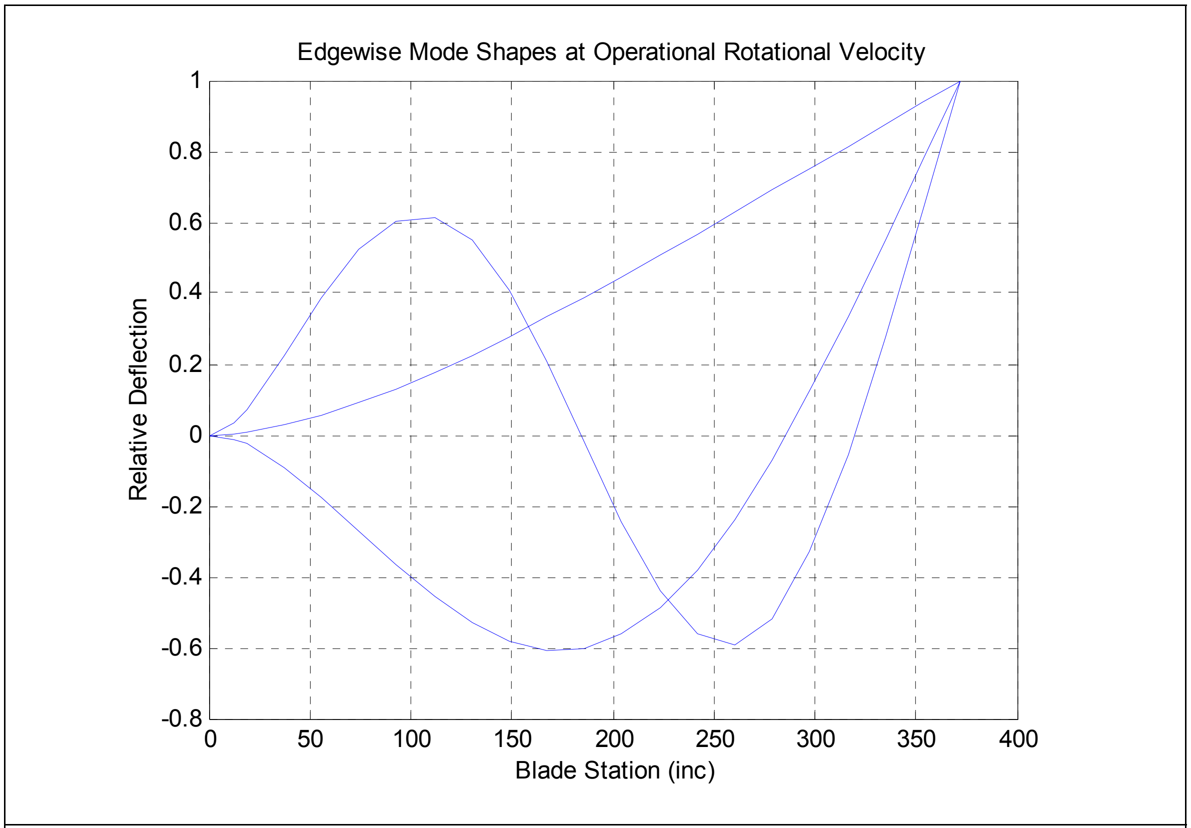
<u>The ratio of the Natural Frequency/Rotational Speed is:</u>

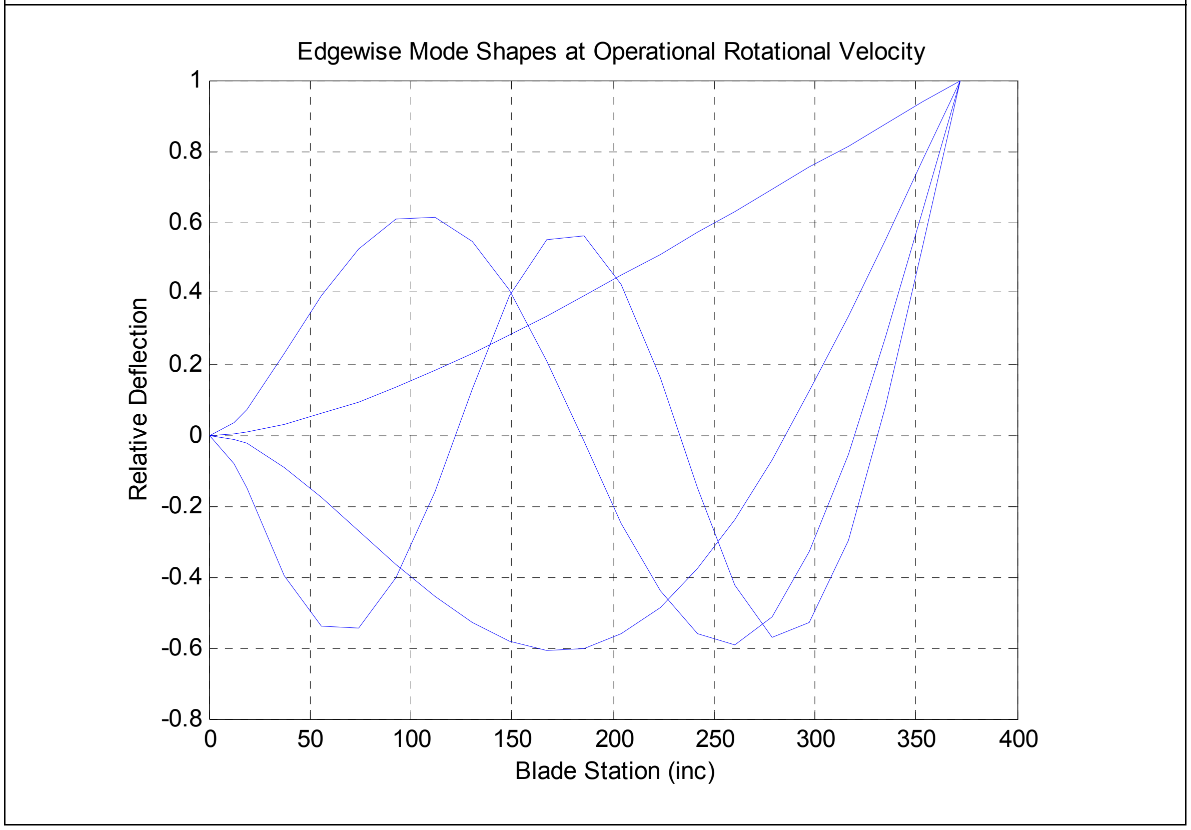
1.12500000000000
2.87500000000000
5.27500000000000
8.02500000000000
11.32500000000000

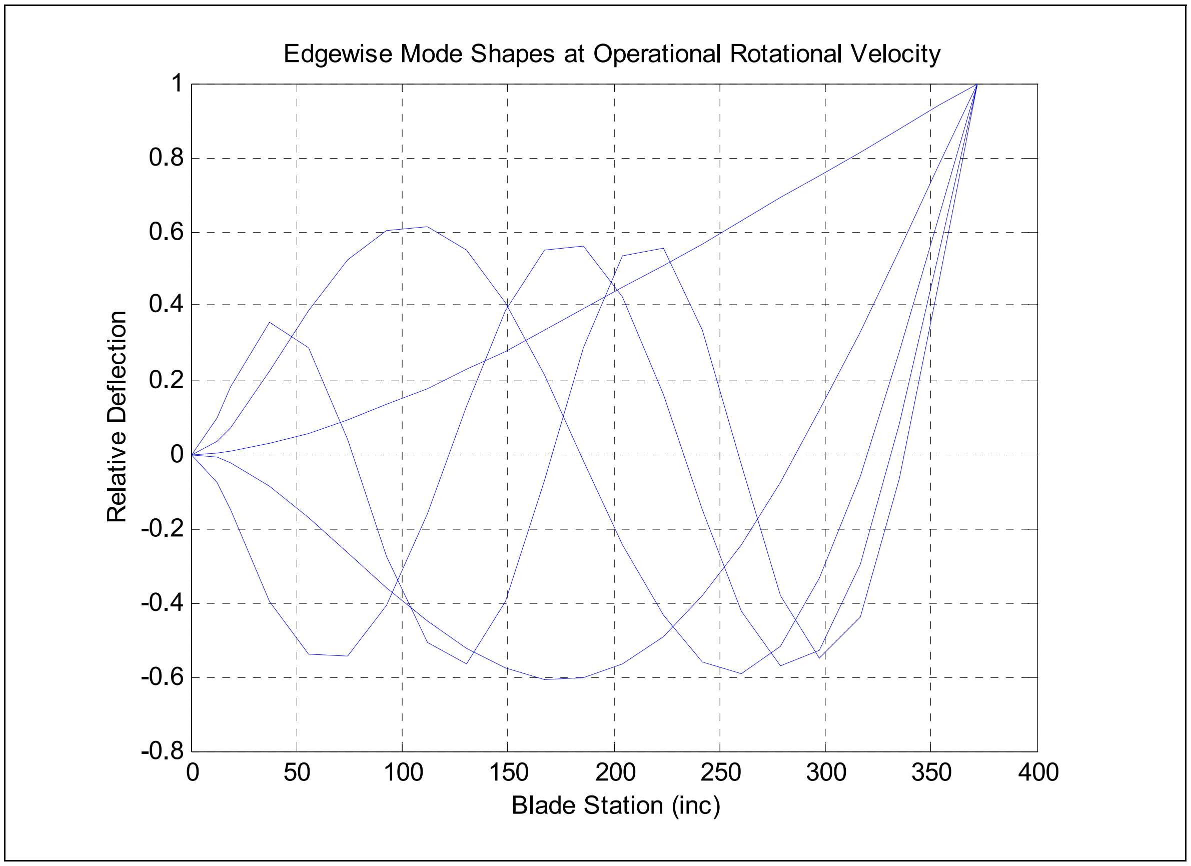
2. Edgewise Mode Shapes






Edgewise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inc)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
50
100
150
200
250
300
350
400


Edgewise Mode Shapes at Operational Rotational Velocity
Relative Deflection
Blade Station (inc)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
50
100
150
200
250
300
350
400



<u>Centrifugal Force :</u>
4.515809216498805e+004

<u>The Natural Frequencies are(cpm):</u>

```
147.17500000000
847.52500000000
2055.37500000000
3598.17500000000
5506.37500000000
```

<u>The ratio of the Natural Frequency/Rotational Speed is:</u>

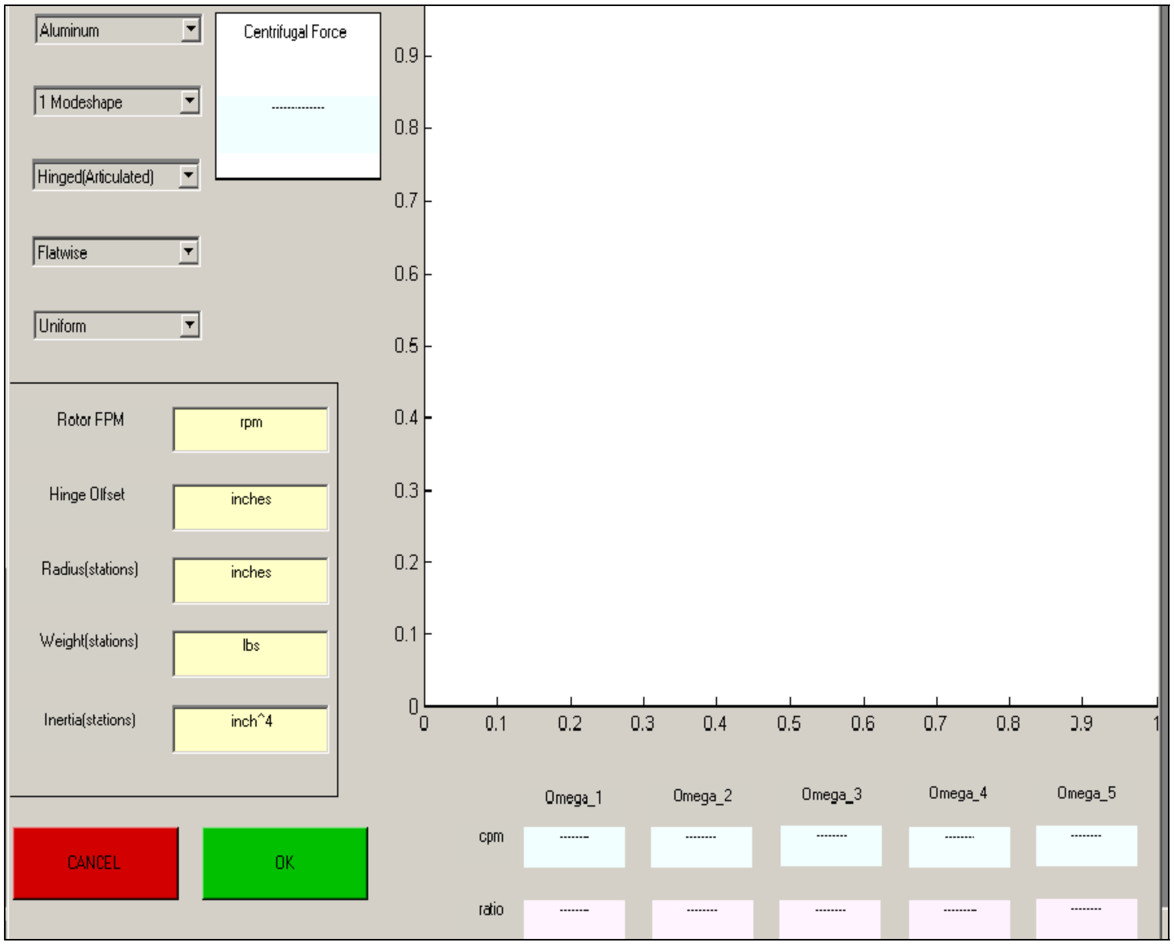
```
0.72500000000000
4.17500000000000
10.12500000000000
17.72500000000000
27.12500000000000
```

APPENDIX D. BUILDING GUI AND INCORPORATING MATLAB® CODE INTO JANRAD

A. GENERATION OF THE GUI

The development of the GUI is accomplished utilizing tools of MATLAB® version 6.1[Ref. 16]. The most important tool in order to generate a GUI is a function called GUIDE (Graphical User Interface Design Environment) allows the programmer to create an interactive window which has the same properties as in Windows, using 'drag and drop' controls or objects from a master pallet. Once the front page of the GUI with a fig-file is generated, MATLAB® version 6.1 creates an m-file code which essentially runs the GUI window. The properties of the each object can be arranged by 'Editor Preferences' tool. Main role of the programmer is to write the m-files and callback functions of the m-file of GUI in order to have the program run as desired. Being the blade dynamics portion of the JANRAD program, this GUI is generated with the newest version of MATLAB®. Different portions and versions of the GUI are outlined by Lapacik [Ref. 7] and Hucke [Ref. 8].

In this thesis GUI page is designed in a different approach that the input icons and axis of the output plot and the results of natural frequencies and the centrifugal force are displayed in one end page. The main reason of such a different approach is to give the opportunity of seeing both the inputs and the results and the plots of the blade dynamics in one window. The user window of the GUI is shown in figure below.



Aluminum
1 Modeshape
Hinged(Articulated)
Flatwise
Uniform

Rotor FPM rpm
Hinge Offset inches
Radius(stations) inches
Weight(stations) lbs
Inertia(stations) inch^4

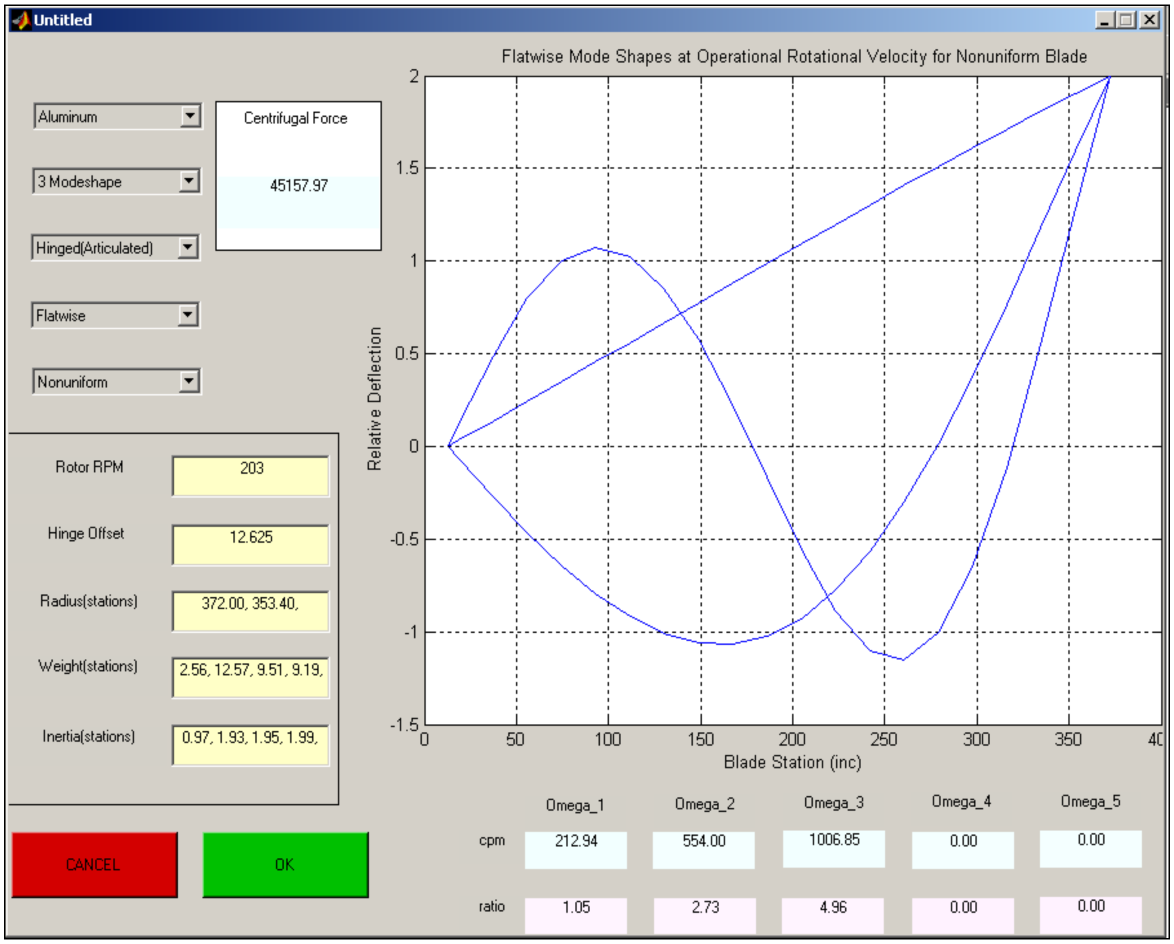
	Omega_1	Omega_2	Omega_3	Omega_4	Omega_5
cpm	-------	-------	-------	-------	-------
ratio	-------	-------	-------	-------	-------

Graphical User Interface (GUI) of the Program

The User’s Guide attached in APPENDIX.E gives details and features of the generated program. For further detailed GUI developments in MATLAB® refer to [Ref. 19] and [Ref. 20].

B. THE RESULTS IN GUI

To show the results in the GUI page, same data for H-3 (S-61) helicopter data is utilized. The results of the application are depicted in figure below. As it can be concluded from the figure the results and the plots of the GUI matches with original generated program in Chapter III. This shows that the developed GUI program is working properly.



Aluminum

3 Modeshape

Hinged(Articulated)

Flatwise

Nonuniform

Centrifugal Force 45157.97

Rotor RPM	203
Hinge Offset	12.625
Radius(stations)	372.00, 353.40,
Weight(stations)	2.56, 12.57, 9.51, 9.19,
Inertia(stations)	0.97, 1.93, 1.95, 1.99,

	Omega_1	Omega_2	Omega_3	Omega_4	Omega_5
cpm	212.94	554.00	1006.85	0.00	0.00
ratio	1.05	2.73	4.96	0.00	0.00

Graphical User Interface (GUI) Plot for H-3 (S-61) Helicopter 'Flat wise'

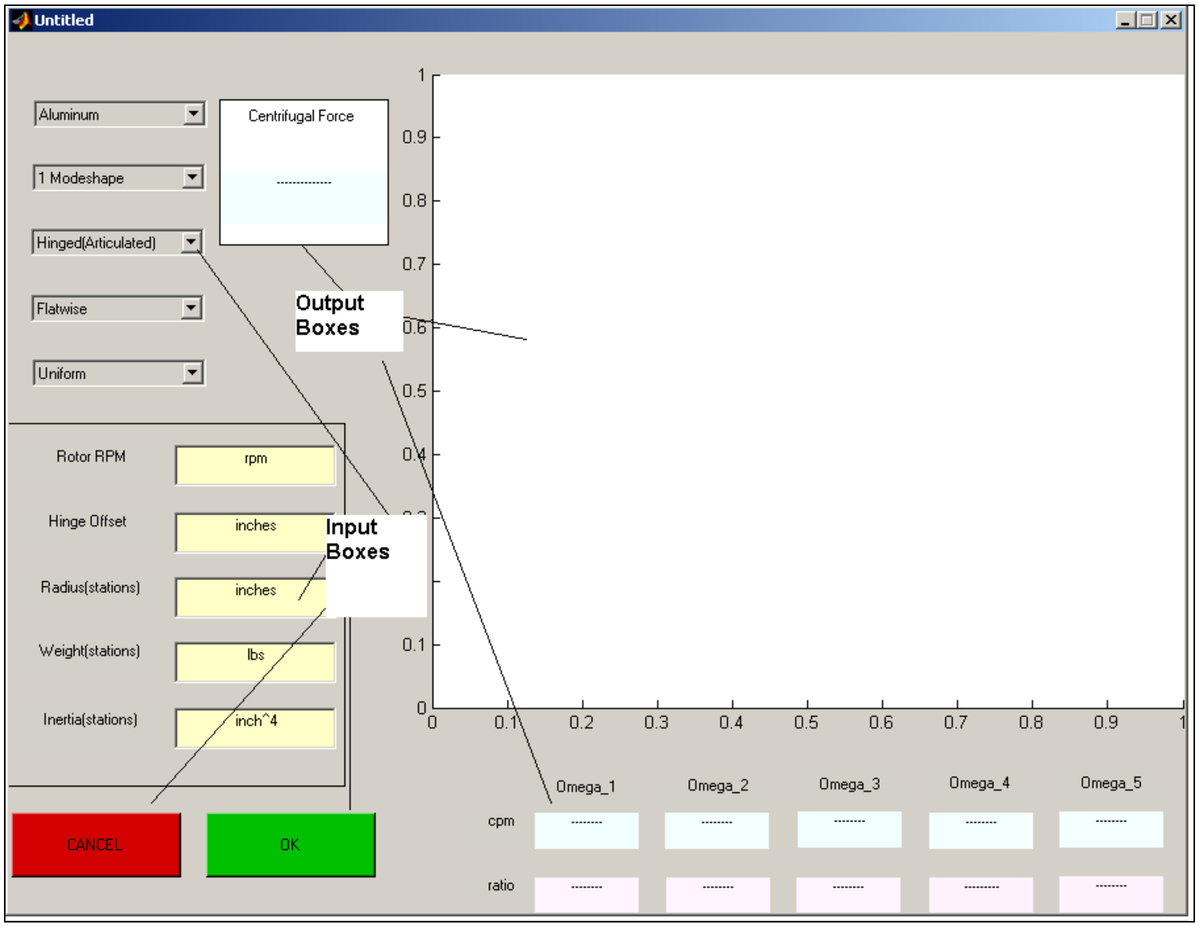
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. USER GUIDE FOR GRAPHICAL USER INTERFACE (GUI)

A. ABOUT THE GUI

GUI window has two essential parts:

- Input Parts
- Output Parts



Main GUI Window

For better understanding refer to the figure above and go through the input and output explanations.

1. Input Boxes

Inputs in this GUI are three types.

a. Popupmenus

There are 5 popupmenus. They are itemized as they are appeared in 'Main GUI Window'.

i. **E_n**

(Elasticity module-stiffness)

You can pick either of the following options:

1. ***Aluminum***
2. ***Composite***
3. ***Titanium***
4. ***Steel***

ii. **Mode shapes**

(Number of mode shapes)

You can select up to 5 mode shapes. If you want to plot more than 5 mode shapes, you should use the related MATLAB® code listed in APPENDIX A manually.

1. ***1.Mode shape***
2. ***2.Mode shape***
3. ***3.Mode shape***
4. ***4.Mode shape***
5. ***5 Mode shape***

iii. **Blade Root**

You can select one of these 2 options:

1. ***Hinged (Articulated) Blade***
2. ***Hingeless (Rigid Blade)***

iv. **Plot Axis**

You can select one of the 3 options:

(For current version of the generated GUI program coupled is not available.)

1. ***Flatwise***
2. ***Edgewise***
3. ***Coupled***

v. **Blade Form**

You can select one of the 2 options.

Note that once you select 'Uniform' blade you should use corresponding values for uniform otherwise the program will not run and give error or warning dialog boxes. It is the same for the nonuniform blade.

1. ***Uniform***
2. ***Nonuniform***

b. Edit Boxes

Write numerical data in the input edit boxes. Do not use any non-numerical inputs; this will avoid the GUI to run, and you will see error statements in MATLAB® Command Window. If the design is about a 'Nonuniform Blade', then the number of the stations should match in the input boxes for " Radius", "Weight" and "Inertia".

i. **Rotor RPM**

Rotational Speed of the design helicopter in units of '*rpm*'.

Do not use '*0*' for rotational speed this will cause problems in GUI.

ii. **Hinge Offset**

Hinge Offset (*e*) of the rotor blade in units of '*inches*'.

You can enter a non-zero value as long as you have picked 'Hinged (Articulated)' in the 'Blade Root' popupmenu box. Otherwise you should enter 0 for 'Hingeless (Rigid)'.

iii. **Radius (stations)**

Radius input changes depending on the selection made in 'Blade Form' popupmenu box.

If you have selected "Uniform": You should enter only one numerical value, which should be the length of the blade in units of '*inches*'.

Use commas (,) in between the stations.

Example:

If the radius values of the blade are: {250, 240, 235, 230,……,10}

Then enter the radius in the edit box as:

250,240,235,230, …………………..,10

iv. **Weight (stations)**

Weight input procedure is the same as the radius procedure. Use the unit of '*lbs*'

Example:

If the weight values of each station are: { 7.6, 3.4, 5, 2,5……, 3}

Then enter the weight in the edit box as:

7.6, 3.4, 5, 2,5 …………………….,3

v. **I_xx (stations)**

Inertia input procedure is the same as the radius procedure. Use the unit of '*inch*^{4}'

Example:

If the inertia values of each station are: { 1.2, 2, 2,5 ,4, …,1,5}

Then enter the inertia in the edit box as:

1.2, 2, 2.5, 5, 4, ………,1.5

c. Push Button

i. **OK**

This pushbutton is used to run the program. Once you entered all the inputs push on this button. After pushing this button GUI will automatically run all the necessary programs, and give the plots and the results on the 'Main GUI Window'.

ii. **CANCEL**

If you push on this it will cancel to run the program.

2. **Output Boxes**

There are two types of outputs in this GUI program.

a. Static Boxes

Note that the inputs entered, still stays in the GUI window after running the program. Hence, you can see the input and output values at the same time.

i. **Centrifugal Force**

The box under this title gives the Centrifugal Force value of inputs entered

ii. **Omega_1…….Omega_5 for side label "cpm"**

The Natural Frequency values are depicted in these boxes on the right side of the 'cpm' label. Natural frequency values correspond the boxes under each 'omega' box.

iii. **Omega_1…..Omega_5 for the side label "ratio"**

Ratio of Natural Frequencies to the Operational Speed is depicted in these boxes on the right side of the 'ratio' label. Each ratio corresponds the boxes under each 'omega' box.

b. Axis

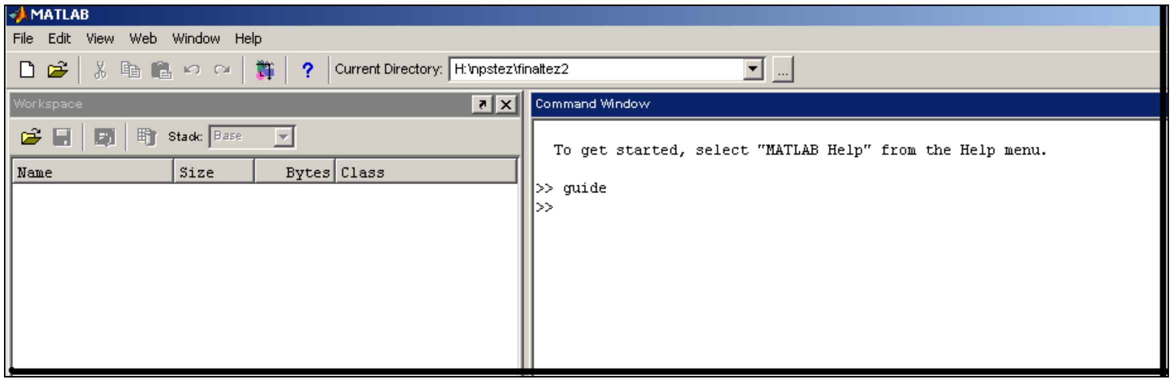
Mode shape plot is depicted in this box. You can see which plot is depicted from both the inputs you have chosen and the title of the plot located in the axis box.

B. RUNNING THE PROGRAM

There are two different ways of running the GUI program:

1. **Using 'guide'command**

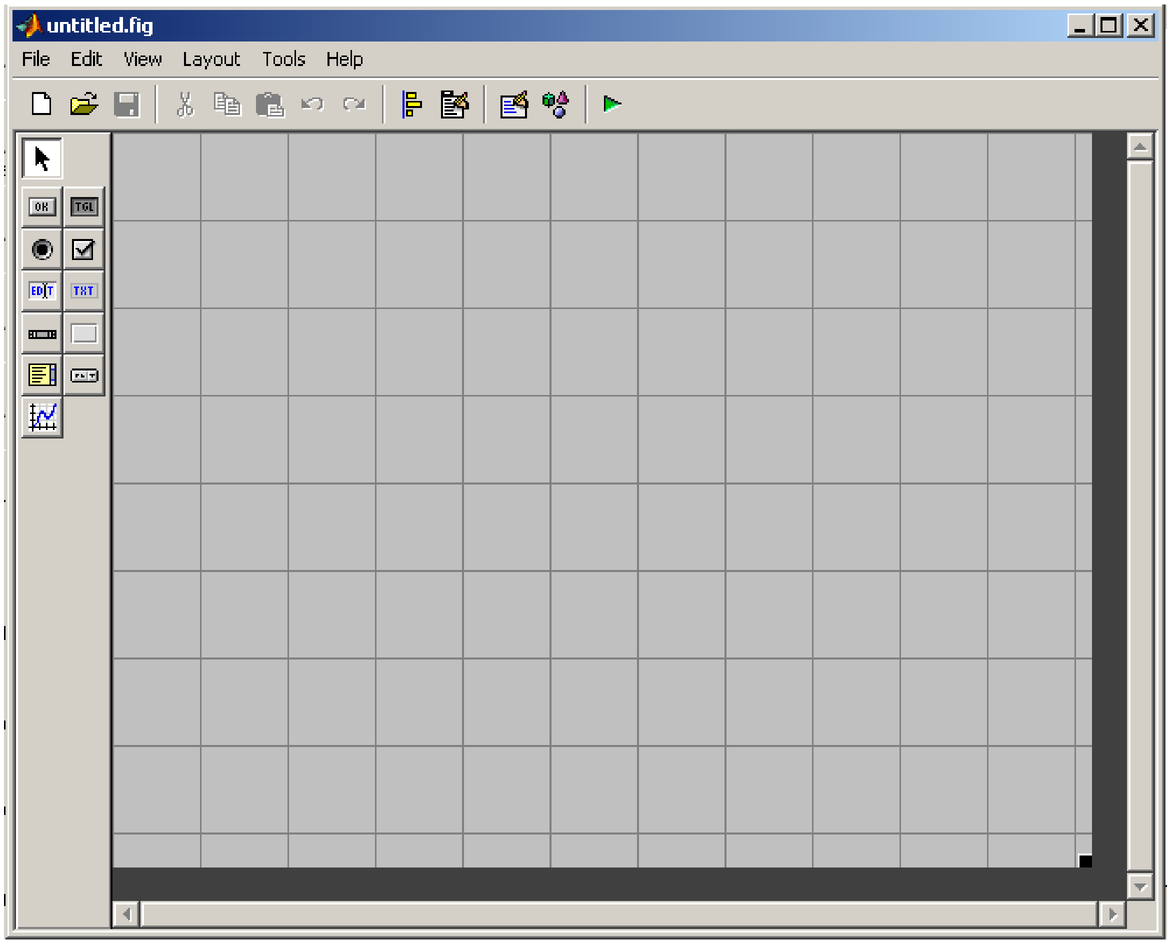
Print 'guide' and press 'enter' in MATLAB® Command window. MATLAB® will open the 'gui layout editor window'


MATLAB
File Edit View Web Window Help
Current Directory: H:\npstez\finaltez2
Workspace
Stack: Base
Name Size Bytes Class
Command Window

```
  To get started, select "MATLAB Help" from the Help menu.

>> guide
>>
```

MATLAB® Command Window



GUI Layout Editor

From this window you can go to 'open files' icon and open the '*finalthesis.fig*' file where it is saved in your computer.

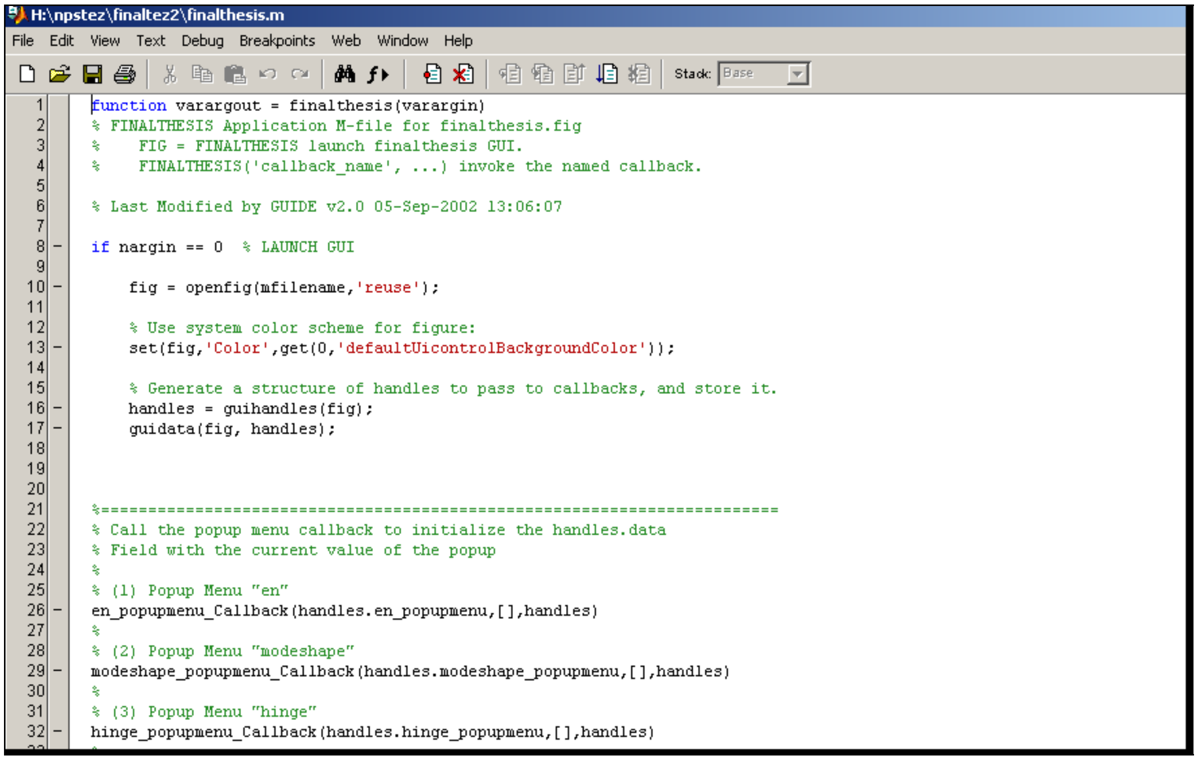
Once you open the '*finalthesis.fig* ' file you will see the similar window as 'Main GUI Window'. But it is not exactly the same window. The window that you have opened is 'gui layout editor window'. Do not make any additions or changes on this window. It may cause problems when you want to run the GUI.

Go to the ▶ on top of the window and click on it. This will open you the 'Main GUI Window'. Then enter the input values as described in previous section and run the program.

2. Running with the m-file

Open the Command window of MATLAB®.

Go to the open files part and open the '*finalthesis.m*'file. This will bring you another window, which has the main running m-file of the GUI. Do not make any changes in this program page. This may cause problems while you are running the GUI


H:\npstez\finaltez2\finalthesis.m
File Edit View Text Debug Breakpoints Web Window Help
Stack: Base

```
function varargout = finalthesis(varargin)
% FINALTHESIS Application M-file for finalthesis.fig
%    FIG = FINALTHESIS launch finalthesis GUI.
%    FINALTHESIS('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 05-Sep-2002 13:06:07

if nargin == 0  % LAUNCH GUI

    fig = openfig(mfilename,'reuse');

    % Use system color scheme for figure:
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);



%==========================================================================
% Call the popup menu callback to initialize the handles.data
% Field with the current value of the popup
%
% (1) Popup Menu "en"
en_popupmenu_Callback(handles.en_popupmenu,[],handles)
%
% (2) Popup Menu "modeshape"
modeshape_popupmenu_Callback(handles.modeshape_popupmenu,[],handles)
%
% (3) Popup Menu "hinge"
hinge_popupmenu_Callback(handles.hinge_popupmenu,[],handles)
```

'finalthesis.m file'

Go to 'debug' and 'run' the program. This will open you the 'main GUI window'.

Then enter the input values as described above and run the program

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Nicholson, Jr.R.K., *Computer Code for Interactive Rotorcraft Preliminary Design Using a Harmonic Balance Method for Rotor Trim*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1993.

2. Wirth, Jr.W.M., *Linear Modeling of Rotorcraft for Stability Analysis and Preliminary Design*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1993.

3. Cuesta, J.D., *Modeling Helicopter Blade Dynamics Using a Modified Myklestad-Prohl Transfer Matrix Method*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1994.

4. Hiatt, D.S., *A Study of Helicopter Rotor Dynamics and Modeling Methods*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

5. Eccles, D.M., *A Validation of the Joint Army/Navy Rotorcraft Analysis and Design Software by Comparison With H-34 and UH-60A Flight Test*, Master's Thesis, Monterey, California, December 1995.

6. Klein, G.D., *Linear Modeling of Tiltrotor Aircraft (In Helicopter and Airplane Modes) for Stability Analysis and Preliminary Design*, Master's Thesis, Monterey, California, June 1996.

7. Lapacik, C.F., *Development of Graphical User Interface (GUI) for Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) Software*, Master's Thesis, Monterey, California, March 1998.

8. Hucke, W.L., *Performance Enhancements to Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) Software and Graphical User Interface (GUI)*, Master's Thesis, Monterey, California, June 1998

9. Mcewen, M.D., *Dynamic System Identification and Modeling of a Rotary Wing UAV for Stability and Control Analysis*, Master's Thesis, Monterey, California, June 1998.

10. Heathorn, D.A., *Stability and Control Module for Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) Software and Graphical User Interface (GUI)*, Master's Thesis, Monterey, California, March 1999.

11. Wood, E.R., "An Introduction to Helicopter Dynamics," paper presented AA 3402 Helicopter Aeromechanics Course, Monterey, California, 5 October 2001.

12. Gerstenberger, W. and Wood, E.R., "Analysis of Helicopter Aerolastic Characteristic in High-Speed Flight," *AIAA Journal*, vol. 1,no.10, pp. 2366-2381, October 1963.

13. Scanlan, R. H. and Rosenbaum R., *Introduction to the Study of Aircraft Vibration and Flutter*, p. 149, The Macmillan Company, 1951

14. Hartog, J.P.D., *Mechanical Vibrations*, pp. 185-433, Dover Publication, Inc., 1985.

15. Bisplinghoff, R.L., Ashley, H., and Halfman, R.L., *Aerolasticity*, pp. 159-188, Dover Publications, Inc., 1996.

16. Chapman, S.J., *MATLAB® Programming for Engineers*, 2d ed., Brooks/Cole, 2002.

17. TRECOM Technical Report 64-15, *Parametric Investigation of the Aerodynamic and Aerolastic Characteristic of Articulated and Rigid (Hingeless) Helicopter Rotor Systems*, by E.R.Wood and A.C. Buffalano, April 1964.

18. Young, D., and Felgar, Jr.R.P., "Tables of Characteristic Functions Representing Normal Modes of Vibration of a Beam," *The University of Texas Publication*, v. 44, no. 4913, pp. 1-33, 1 July 1949.

19. Marchand, P., *Graphics and GUIs with MATLAB*, 2d ed., CRC Press, 1999.

20. Chapman, S.J., *MATLAB® Programming for Engineers*, Brooks/Cole, 2000.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Kara Kuvveleri Komutanligi
   Ankara, Turkiye

4. K.H.O. Savunma Bilimleri Enstitusu
   Ankara, Turkiye

5. Professor E. Roberts Wood
   Naval Postgraduate School
   Monterey, California

6. CDR Mark A. Couch
   Naval Postgraduate School
   Monterey, California

7. I.T.U. Ucak ve Uzay Bilimleri Fakultesi Dekanligi
   Maslak, Istanbul, Turkiye

8. O.D.T.U. Havacilik Muhendisligi Bolumu
   Ankara, Turkiye

9. Anadolu Universitesi Sivil Havacilik Okulu
   Eskisehir, Turkiye

10. Erciyes Universitesi Sivil Havacilik Okulu
    Kayseri, Turkiye

11. Turkish Aerospace Industries Inc.
    Ankara, Turkiye

12. 901.Hv.Arc.A.Dp. ve Fb. K.ligi
    Guvercinlik, Ankara, Turkiye

13. 1 LT. Dogan Ozturk
    Turkish Army
    Ankara, Turkiye

14. 1 LT. Hakki Erdem Akin
    Turkish Army
    Ankara, Turkiye

15. Fikri Ozturk
    Manisa, Turkiye

16. Refik Irfan Ozturk
    Istanbul, Turkiye

17. Suat Gokhan Ozturk
    Belleville, New Jersey

18. Iskender Girgin
    Ankara, Turkiye